SF 298 MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

# REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188



| 1 AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | December 1997 | Technical 97-24 |

**4 TITLE AND SUBTITLE**
Fitting Optimal Piecewise Linear Functions Using Genetic Algorithms

**5. FUNDING NUMBERS**
DAAH04-96-1-0082

**6 AUTHOR(S)**
Jennifer Pittman and C.A. Murthy

**7 PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)**
Center for Multivariate Analysis
Department of Statistics
417 Thomas Bldg.
Penn State University
University Park, PA 16802

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9 SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

**10 SPONSORING / MONITORING AGENCY REPORT NUMBER**
ARO 35518.26-MA

**11 SUPPLEMENTARY NOTES**
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**12 b. DISTRIBUTION CODE**

**13 ABSTRACT**

Constructing a model for data in $\mathcal{R}^2$ is a common problem in many scientific fields, including pattern recognition, computer vision, and applied mathematics. Often, little is known about the process which generated the data or its statistical properties. For example, in fitting a piecewise linear model the number of pieces as well as the knot locations may be unknown. Hence the method used to build the statistical model should have few assumptions and yet still provide a model that is optimal in some sense. Such methods can be designed through the use of genetic algorithms.

In this paper we examine the use of genetic algorithms to fit piecewise linear functions to data in $\mathcal{R}^2$. The number of pieces, the location of the knots, and the underlying distribution of the data are assumed to be unknown. We discuss existing methods which attempt to solve this problem and introduce a new method which employs genetic algorithms to optimize the number and location of the linear pieces. We prove theoretically that our method provides near-optimal functions and present the results of extensive expreiments which demonstrate that the proposed method provides better results than existing spline based methods. We conclude that our method represents a valuable tool for fitting both robust and non-robust piecewise linear functions.

**14 SUBJECT TERMS**
Genetic Algorithms, Optimization, Statistical Data Analysis, Splines, Neural Networks

**15. NUMBER IF PAGES**
40

**16. PRICE CODE**

| 17 SECURITY CLASSIFICATION OR REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

# FITTING OPTIMAL PIECEWISE LINEAR FUNCTIONS USING GENETIC ALGORITHMS

**Jennifer Pittman and C.A. Murthy**

Technical Report 97-24

December 1997

Center for Multivariate Analysis
417 Thomas Building
Penn State University
University Park, PA 16802



Research work of authors was supported by the Army Research Office under Grant DAAHO4-96-1-0082. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

# Fitting Optimal Piecewise Linear Functions Using Genetic Algorithms

Jennifer Pittman and C. A. Murthy*

Center for Multivariate Analysis
Department of Statistics
326 Thomas Building
Pennsylvania State University
University Park, PA - 16802, USA

December 4, 1997

*On lien from the Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta - 700035, India.

## Abstract

Constructing a model for data in $\mathcal{R}^2$ is a common problem in many scientific fields, including pattern recognition, computer vision, and applied mathematics. Often, little is known about the process which generated the data or its statistical properties. For example, in fitting a piecewise linear model the number of pieces as well as the knot locations may be unknown. Hence the method used to build the statistical model should have few assumptions and yet still provide a model that is optimal in some sense. Such methods can be designed through the use of genetic algorithms.

In this paper we examine the use of genetic algorithms to fit piecewise linear functions to data in $\mathcal{R}^2$. The number of pieces, the location of the knots, and the underlying distribution of the data are assumed to be unknown. We discuss existing methods which attempt to solve this problem and introduce a new method which employs genetic algorithms to optimize the number and location of the linear pieces. We prove theoretically that our method provides near-optimal functions and present the results of extensive expreiments which demonstrate that the proposed method provides better results than existing spline based methods. We conclude that our method represents a valuable tool for fitting both robust and non-robust piecewise linear functions.



**Key Words**: Genetic Algorithms, Optimization, Statistical Data Analysis, Splines, Neural Networks

# 1 Introduction

One of the purposes of statistical data analysis is to determine a functional relationship between input and output variables given a dataset of noisy observations. The dataset, denoted as $D = \{(\mathrm{x}_i, \mathrm{y}_i),\ i = 1, \ldots, N\}$, is assumed to be a number of realizations of some underlying process combined with random noise, i.e.

$$y = f(x) + \epsilon$$

where $\epsilon$ has mean zero. The problem of determining $f(x)$ given a set of data points and various assumptions is relevant to many application fields including engineering, chemometrics, and materials science [1, 2].

The construction of a functional model for $f(x)$, denoted by $\hat{f}(x)$, can involve classical statistical tools such as kernel methods, regression, and splines [3, 4], as well as more recent techniques such as neural networks, radial basis functions, and genetic algorithms [5, 6, 8]. In using these tools the function $f(x)$ is often assumed to be of a certain form and to meet certain mathematical requirements, such as continuity or differentiability. In $\mathcal{R}^2$, the assumed form of $f(x)$ is usually linear, piecewise-linear, or curvilinear. Since the linear models represent a subset of the piecewise-linear models, and any curve can be approximated by a piecewise-linear function, we chose to fit piecewise-linear models to our datasets. In this case, we cannot assume strict differentiability. However, usually it is the choice of technique, rather than the data or prior process knowledge, that motivates the placement of artificial mathematical restrictions on the final model [9]. It should also be noted that not all techniques can guarantee convergence to a near-optimal $\hat{f}(x)$. Given these considerations and our desire to optimize the number of pieces as well as their placement in the model, we utilized genetic algorithms as our primary model fitting tool.

Genetic algorithms (GAs) are stochastic optimization tools from the field of artificial intelligence [11]. They are capable of finding near-optimal solutions to problems without the usual mathematical model restrictions [9, 10]. In this work we discuss how genetic algorithms were employed to fit piecewise linear models to data sets in $\mathcal{R}^2$, where the number of pieces (or knots) as well as their placement are unknown. We first present the problem and discuss current methods for function approximation. We then introduce genetic algorithms and detail how GAs can be used to fit near optimal piecewise-linear functions. Several examples are presented with results, comparisons are made to existing methods, and areas for future research are mentioned.

# 2 Problem Statement and Current Methodology

Suppose we are given a data set $D = \{(\mathrm{x}_1, \mathrm{y}_1), (\mathrm{x}_2, \mathrm{y}_2), \ldots, (\mathrm{x}_N, \mathrm{y}_N)\} \subset \mathcal{R}^2$ where the values $(\mathrm{x}_i, \mathrm{y}_i)$ are related by an unknown function $f$ such that $\mathrm{y}_i = f(\mathrm{x}_i) + \epsilon_i$, where $\epsilon_i$ is a random error with zero mean and constant variance. We assume that

- $f$ is an $h^*$-piecewise linear function where $h^*$ is an unknown positive integer representing the number of pieces, $1 \leq h^* \leq h_{\max}$, $h_{\max}$ known.
- The knot locations $(z_{11}, z_{12}), \ldots, (z_{(h^*+1)1}, z_{(h^*+1)2})$ of $f$ are unknown (the endpoints of $f$ are also considered knots).

The problem is to fit a function to the data such that

- The fitted function maximizes the quality of the fit over all such $h$-piecewise linear functions, $1 \leq h \leq h_{\max}$, where the quality of the fit is determined by a specified function $FF$.

$FF$ is defined in such a way that the fitted function is 'close' to $f$. We make no assumptions regarding the smoothness of functions around their knots.

Classical approximation theory suggests several methods for fitting piecewise linear functions, methods which involve building models from linear combinations of nonlinear functions [12]. Such linear estimators can be expressed as

$$\hat{f}(x) = \sum_{i=1}^{n} K_\lambda(x, x_i) y_i \tag{1}$$

where $K_\lambda(x, x_i)$ is a weighting function which depends on some parameter(s) $\lambda$ [3]. Some examples include kernel estimates and series approximators (which we will not explicitly discuss), splines, locally weighted regression, and the more recent neural network and radial basis function estimates.

A *spline* of order L with knots at $(z_{11}, z_{12}), \ldots, (z_{K1}, z_{K2})$ is a function $s$ of the form

$$s(x) = \sum_{i=0}^{L-1} \theta_i x^i + \sum_{i=1}^{K} \delta_i (x - z_{i1})^{(L-1)} \tag{2}$$

for $\theta_i \in \mathcal{R}$, $i = 0, \ldots, L-1$, and $\delta_j \in \mathcal{R}$, $j = 1, \ldots, K$ [13]. In other words, a spline is a piecewise polynomial where the pieces are tied at knots in such a way that $s$ satisfies certain continuity properties (e.g., the first $L-1$ derivatives of $s$ are continuous). As such they can be viewed as an extension of polynomial regression [3]. Different classes of splines can be formed by using different basis functions, e.g., B-splines, periodic splines, etc. Splines are useful when we want an estimate which meets a certain quality of fit (fitness) as well as a smoothness criterion. For example, we may estimate $y$ by choosing $\hat{f}$ to minimize

$$n^{-1} \sum_{i=1}^{N} (y - \hat{f}(x_i))^2 + \lambda \int_a^b \hat{f}^{(m)} dx \tag{3}$$

for $\lambda > 0$, $m \in \{1, 2, 3, \ldots\}$, and $a \leq x_i \leq b \; \forall \; i = 1, \ldots, N$. The solution $\hat{f}$ is called a *smoothing spline* estimate, and $\lambda$ is the *smoothing parameter*. $\lambda$ determines the tradeoff between goodness-of-fit and smoothness. If, however, our fitness criterion is not of this

form, a spline may not be the best type of estimate. Splines have applications in areas such as materials science [1], computer tomography [3], and military analysis [14].

To use (smoothing) splines for analysis, the order $L$ of the spline, the number and location of the knots, the smoothing parameter $\lambda$, the choice of basis, and the smoothing criterion need to be specified. This may not be a simple task. How to determine 'optimal values' for these parameters is an area of active research. Methods designed to find a good estimate for $\lambda$ are computationally demanding [15] with GCV estimates tending to oversmooth the data [16]. $m$ is often based on prior information [3] as opposed to theoretical considerations. Schwetlick and Schütze [19] describe an algorithm which optimizes the location and number of 'free' knots but is computationally 'too expensive' to implement and involves the approximation of various parameters whose effects on the final estimate are unknown. Larson [14] finds a closed form for the minimizing abscissa for unknown knot locations of an interpolating spline, but does not mention the optimization of the number of knots. Suchomski [20] and Kokkonis [1] have done some work with fitting least squares variable knot interpolating splines with the number of knots fixed. Mammen [4] addresses the stated problem with locally adaptive regression splines, finding the best fit for all $\lambda$ given a fixed order $L$ but with knots restricted to the class of data points.

Locally weighted regression, or LOESS [21], fits an estimate $g(x)$ which uses a neighborhood of weighted observations whose values $\mathrm{x}_i$ are closest to $x$. $g(x)$ may be written as

$$g(x) = \sum_{i=1}^{N} l_i(x)\mathrm{y}_i \tag{4}$$

where the $l_i(x)$, $i = 1, \ldots, N$, depend on $\{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{N}\}$, a neighborhood size $q$, a weighting function $W$, and a distance measure $p$. Local fitting allows the estimation of a wide class of regression surfaces, wider than that which can be estimated by polynomials. However, LOESS does assume that $y$ has a normal distribution and that the estimate $g(x)$ is unbiased.

A recent development in functional approximation is the use of *neural networks* (NN) and *radial basis functions* (RBFs). Multilayer neural networks are function approximators (in the sense of (1)) of the form, e.g.,

$$\hat{f}(x, \mathcal{W}) = \sum_{j=1}^{M} \beta_j g_j(\alpha_j x) \tag{5}$$

where $\beta_j, 1 \leq j \leq M$, are the weights connecting $M$ hidden units to the output unit, $\alpha_j, 1 \leq j \leq M$, are weights connecting the input layer unit to the $j$th hidden layer unit, and the $g_j$'s are the hidden layer activation functions [12]. $\mathcal{W}$ represents the matrix of network weights. Because of the form of $\hat{f}$, splines and kernel estimates are sometimes viewed as special cases of NN models. Another special case of NN models is based on radial basis functions where the approximation is produced by passing each $\mathrm{x}_i$ through a set of basis functions (each containing a RBF center), multiplying the result by a coefficient, and then summing the results. In other words,

$$\hat{f}(\mathrm{x}_k, \mathcal{W}) = w_0 + \sum_{j=1}^{M} w_j \tau^{-p} \phi(\|x - c_j\|/\tau) \tag{6}$$

where $\phi$ is the radial basis function, $\{c_j : j = 1, \ldots, M\}$ is the set of RBF centers, $\tau$ is a scale parameter, and $p$ is the dimension of the data. Often $\phi$ corresponds to a Gaussian density [5, 6]. Note that a radial basis function network (RBFN) is essentially a kernel method for regression. In general, NNs are easily programmed and, as a result, have become an almost universal optimization 'crank': simply toss in the data, add any number of parameters, and wait for gradient descent to produce the result [22]. However, there is no method for finding the best network architecture for a given problem. Once an architecture has been chosen (heuristically), the system often requires numerous adjustments, supplied by an experienced user, and do have a tendency to overfit or overparameterize the data [22]. They may also get stuck at local minima, unlike GAs [5, 25]. Chen and Jain [12] report that backward propagation can be slow and sensitive to noise. They suggest a robust modification whose parameters are the focus of further study. RBFNs have been shown to outperform multilayer perceptrons (MLPs) [6] even though the choice of centers and the curse of dimensionality can make implementation difficult. It should also be noted that both NN and RBFN results lack interpretability [22].

A discussion of the advantages and disadvantages of the above methods led us to consider the possibility of using genetic algorithms (GAs) as a data fitting tool. GAs need three things to construct a model for a dataset: a way to calculate a response, an error function (i.e., a measure of quality of fit), and a set of parameter values. However, unlike splines, the values given to the parameters are not critical to the algorithm's result or its convergence - they only influence the rate at which the algorithm converges. By utilizing GAs we may discard the artificial requirements of differentiability and smoothness imposed on the functional form of the model by the above methods [9]. It is also possible to define the fitness (or optimization) function in such a way that the same algorithm can be used to fit both robust and non-robust functions - robust in the sense of being insensitive to outliers. In the case of piecewise linear models, the power of GA optimization allows not only variable knot placement but also a variable number of knots. Such variable models have been fit using Bayesian estimation as opposed to GA optimization [23, 24], but GAs can guarantee convergence to an optimal solution for a sufficiently large number of iterations [25]. Related works include those of Karr [2, 8], who has applied GAs to the problem of least squares (LS) and least median squares (LMS) curve fitting where the specific form of the curve is known (e.g., a polynomial of degree 2) and the data is noiseless, and Vankeerberghen [26], who have used GAs to obtain the parameters for specific, known laboratory system models (e.g., a hyperbolic model) which are nonlinear in the parameters. We intend to design a GA based method which fits both robust and non-robust near-optimal piecewise linear functions to data in $\mathcal{R}^2$.

# 3 Genetic Algorithms

*Genetic algorithms* are stochastic search methods which provide a near optimal solution to the evaluation function of an optimization problem [9]. They can be used to search complex, multimodal surfaces via steps which have been designed to mimic the processes of natural genetic systems. They work simultaneously on a number of possible solutions to help prevent the algorithm from getting stuck in a local optimum.

With little adjustment - in most cases by only redefining the fitness function - GAs can be applied to a wide range of optimization problems. Thus it is possible to use the same basic algorithm to, for example, fit lines satisfying different optimization criterions to the same dataset. Situations in which the effectiveness of GAs has been demonstrated can be found in the literature of various scientific fields such as scheduling, classifier systems, and pattern recognition [10].

In GA algorithms, each possible solution is encoded as a string or chromosome; a set of such chromosomes is called a *population.* An evaluation (*fitness*) function provides a mapping from the chromosome space to the solution space. The algorithm starts with an initial population of a fixed number of randomly generated strings. At each iteration, three basic operations - selection, crossover, and mutation - are applied over the current population to yield a new population of strings. This cycle is repeated until some termination criterion is met, at which time the best string achieved is generally taken as the solution to the optimization problem.

We will briefly outline the basic framework of a GA before discussing specific GA concepts that can be applied to our problem.

## 3.1 Basic Framework

For a more detailed look at the GA framework, we will discuss the stages of a specific GA model, the *elitist* model. Consider the problem of maximizing a function $FF(x) \in V$, where $V$ is a finite set and $FF(x) > 0 \; \forall \; x \in V$. Each string $S$, built from members of a finite alphabet $\mathcal{A} = \{\alpha_1, \ldots, \alpha_a\}$, corresponds to a value $x$ in $V$ and may be written as

$$S = (\gamma_1, \gamma_2, \ldots, \gamma_{lgth}); \;\; \gamma_i \in \mathcal{A} \;\; \forall \, i = 1, \ldots, lgth$$

The number of different strings that are possible is $a^{lgth}$. For our work we will be using binary strings, i.e., $\mathcal{A} = \{0, 1\}$, so $a^{lgth} = 2^{lgth}$. A random sample of size $M$ ($M$ even) is drawn from these possible strings with replacement to form the initial population $\mathcal{Q}$. The evaluation or fitness value of each string $S$ is $fit(S) = FF(x)$ where $x \in V$ is the value represented by $S$.

The first operation, *selection*, is modeled after Darwin's concept of 'survival of the fittest'. Strings from the population are selected and placed in a mating pool; the probability of selection for string j is $p_s^{(j)} = fit(S_j) / \sum_{i=1}^{M} fit(S_i)$. For example, if $P_s^{(j)} = \sum_{i=1}^{j} p_s^{(i)}$, $M$ strings are selected and placed in the mating pool by repeating the following process $M$ times:

1. Generate a random number $rnd_i$ from [0,1]
2. If $rnd_i \leq P_s^{(1)}$, select $S_1$; for $j = 2, \ldots, M$, if $P_s^{(j-1)} < rnd_i \leq P_s^{(j)}$, select $S_j$

Note that strings with low fitness values are rarely selected while some strings may be selected more than once. The mating pool is taken as our new population $\mathcal{Q}_1$.

In *single point crossover*, or *reproduction*, pairs of strings exchange information, thereby generating two new offspring for the next population. All strings are paired at random

in such a way that each string belongs to only one pair (hence there are $M/2$ pairs). If the given pair is denoted as

$$\beta = (\beta_1, \ldots, \beta_{lgth}) \text{ and } \tau = (\tau_1, \ldots, \tau_{lgth})$$

and $p_c$ is the probability that a given pair of strings undergoes crossover, then the crossover operation may be described as

1. Generate a random number $rnd$ from [0,1]
2. If $rnd > p_c$, no crossover is performed. If $rnd \leq p_c$,
   (a) Generate a random integer $pos$ from [1,$lgth$-1].
   (b) Strings $\beta$ and $\tau$ are replaced by strings $\beta'$ and $\tau'$ where

$$\beta' = (\beta_1, \ldots, \beta_{pos}, \tau_{pos+1}, \ldots, \tau_{lgth}) \text{ and } \tau' = (\tau_1, \ldots, \tau_{pos}, \beta_{pos+1}, \ldots, \beta_{lgth})$$

Once this operation has been applied to each pair of strings, the resulting population is denoted $\mathcal{Q}_2$. $\mathcal{Q}_2$ has $M$ strings, some of which may have also been elements of $\mathcal{Q}_1$.

*Mutation* involves the random altering of characters in the strings of $\mathcal{Q}_2$. Let $p_m$ denote the probability of mutation of a given character. Then, for each character $\beta_i$ of every string $\beta$, the mutation stage consists of

1. Generate a random number $rnd$ from [0,1]
2. If $rnd > p_m$, $\beta_i$ is not mutated. If $rnd \leq p_m$,
   (a) If $\beta_i$=0, then $\beta_i$ becomes 1; else, $\beta_i$ becomes 0.

The mutation probability may vary over iterations, initially taking a high value, then decreasing to a pre-specified minimum, then increasing again in the later stages of the algorithm (see [29]). When the algorithm has little knowledge of the search space, it is encouraged to explore its domain through a high mutation probability. As the number of iterations increases the algorithm will move towards a solution, hence the mutation probability is decreased to allow a search in the vicinity of this solution. To avoid the attraction of the algorithm to a local optimum, the mutation probability is increased in the later stages to again allow for a broader search. The resulting population we denote as $\mathcal{Q}_3$.

Note that through mutation, a given string can become any of the $2^{lgth}$ possible strings.

We now replace our initial $\mathcal{Q}$ with $\mathcal{Q}_3$ and repeat the above stages until the algorithm converges to a satisfactory solution. The stages we have discussed so far are common to all GA models. In the elitist model of GA's (EGA), a further operation, *elitism*, is added to ensure that as the algorithm is running knowledge about the best string obtained so far is preserved. In this way the algorithm can report at any time the best solution which has been achieved. The basic steps which are added to the above model to form the elitist model are

1. Once the initial population $Q$ has been generated, find the fitness value of each string $S$ in $Q$. Let the string with the maximum fitness value $fit_{maxQ}$ of all of the strings in $Q$ be denoted as $S_{maxQ}$.

2. Compare the fitness value of each string in $Q_3$ with the fitness value of $S_{maxQ}$. If no string in $Q_3$ has a fitness value greater than or equal to $fit_{maxQ}$, replace the worst string in $Q_3$ with $S_{maxQ}$.

At this point, $Q_3$ is taken as the new $Q$ and the algorithm begins another iteration by returning to the first operation, *selection.*

This basic framework can be modified to specifically address the problem of interest. One such modified algorithm which includes concepts which can be applied to our problem is the *variable length genetic algorithm* (VLGA). These concepts will be used to design an algorithm for our problem whose computational complexity is less than that of the VLGA.

## 3.2 Variable Length Genetic Algorithm (VLGA)

The variable length genetic algorithm was designed by Bandyopadhyay, Murthy, and Pal [27] for optimizing the number of hyperplanes needed to classify patterns in a multidimensional feature space. It is able to consider solutions with varying numbers of planes by using strings of varying lengths. For example, a set of $h$ hyperplanes, $1 \leq h \leq h_{\max}$, $h_{\max}$ known, could be represented by the string

$$S = (\gamma_1, \gamma_2, \ldots, \gamma_{h*L}); \quad \gamma_i \in \mathcal{A} \ \forall \, i = 1, \ldots, h * L$$

where $\mathcal{A}$ is the set of possible characters and $L$ characters (e.g., bits) are used to represent each plane. Hence the string length is not fixed but *variable* - for a given $h \in \mathcal{H} = \{1, \ldots, h_{\max}\}$, the string length is $h * L$. The algorithm included the basic stages - selection, crossover, and mutation - as well as elitism, modified to handle strings of different lengths. The elitism stage ensures that the algorithm will converge to an optimal solution, theoretically.

The fitness criterion was defined so that the set of hyperplanes with the maximum fitness value would (1) have the minimum number of misclassifications of any set of $h$ hyperplanes, $h \in \mathcal{H}$, and (2) use as few hyperplanes as possible in doing so. Hence the fitness function could be represented as

$$fit(S) = (N - miss_S) + \frac{h_S}{h_{\max}} \tag{7}$$

where $N$ is the number of data points, $miss_S$ is the number of points misclassified by the set of hyperplanes represented by $S$, $h_S$ is the number of hyperplanes represented by $S$, and $h_{\max}$ is the maximum possible number of hyperplanes. Hence maximizing the fitness function will yield a set of hyperplanes which is parsimonious yet minimizes the number of misclassifications.

In designing the proposed algorithm we included an elitist stage , as was done in the VLGA, to ensure algorithm convergence to an optimal solution. The more important concept borrowed from the VLGA, however, is the form of the fitness function. The VLGA fitness function is based on two criteria - minimization of the number of misclassifications and the number of hyperplanes. Similarly, we are interested in finding a solution which (1) is 'closest' to the majority of the data points yet (2) does so using the least number of lines possible. Hence the form of the proposed fitness function will mimic that of the VLGA fitness function.

## 3.3 Convergence

Bhandari, Murthy, and Pal [25] have proven theoretically that an elitist genetic algorithm (fixed length strings) will converge to an optimal string as the number of iterations goes to infinity. The two characteristics that are necessary and sufficient for algorithm convergence are

- The optimal string from the present population has a fitness value no less than the fitness values of the optimal strings from the previous populations.
- Each string has a positive probability of going to an optimal string within any given iteration.

By including the elitist stage in the proposed algorithm, we will be able to ensure convergence to an optimal solution. This convergence is independent of the choice of values for the algorithm parameters ($M$, $p_c$, $p_m$) although these parameter values do influence the rate of convergence. There is no theory to indicate the number of iterations necessary for convergence. Two popular heuristic stopping rules are

- Execute the process for a fixed number of iterations and report the best string found as the solution.
- Execute the process until the fitness value does not show adequate improvement over a fixed number of iterations, and report the best string found as the solution.

We will employ the first stopping rule in our GA based method.

## 3.4 Piecewise Linear Fitting

We have discussed the elitist GA and its convergence to an optimal string. This convergence led us to include elitism in our GA based method for finding a near-optimal solution to the problem of fitting an $h$-piecewise linear function to a given dataset $D$ in $\mathcal{R}^2$. Designing such an algorithm required us to 'encode' each possible solution as a string, define the fitness function to reflect the idea of a 'properly' fitted function, and select values for the algorithm parameters ($M, p_c$, etc.). How these steps were achieved will be discussed in Sections 4 and 5.1.1.

We have yet to define the meaning of 'a set of lines representing a distribution' - this will be done in Section 4. This meaning is critical - if the given datapoints are from this distribution, then the representative set of lines is the 'ideal' solution which we seek. Note that the definition of any fitness function will be dependent upon the given dataset $D$. Thus, for the same dataset size $N$, we will get different optimal solutions depending upon the choice of $D$ (assuming that all the steps mentioned above have been achieved) regardless of the specific form of the fitness function. However, if the datasets are from the same distribution, the results obtained by our method should be related. The nature of the relationship between these results needs to be explored in terms of the fitness function, theoretically. Lastly, the fitness function needs to be defined in a way that as the number of iterations and the datset size $N$ go to infinity, the solution obtained will go to the 'ideal' solution.

Sometimes the given dataset may contain 'outliers' which we do not want to influence our determination of the fitness of a given string. Note that the meaning of 'outlier' is unclear, yet we would like to define the proposed fitness function in such a way that the 'outliers' do not contribute to its value. We shall, for the sake of convenience, assume that at most *crit* percent of the data points are 'outliers' and remove them from the fitness calculation. The value $1 - crit$ is referred to as the *critical level.* The value of *crit* depends on the given dataset and/or the user. If the dataset has no 'outliers', then *crit* may be taken as 0.

The GA based optimization method to be proposed can be used for any given value of *crit*. Hence this optimization scheme may be described as **robust** from the point of view of not yielding to the influence of outliers [26, 28]. Theoretically, the properties of the fitness function - where the fitness function is defined with the intention of removing *crit* outliers - are to be studied.

The present article deals with all of these theoretical issues. In the next section, the phrase 'a set of lines representing a continuous distribution' is defined mathematically. A function $FF$, which will later be taken as our fitness function, is also defined. We will be using the polar representation of a straight line for the purpose of coding. This polar representation will also be used in the development of our mathematical theory.

# 4 Theory of Piecewise Linear Fitting in $\mathcal{R}^2$

## 4.1 Mathematical Formulation

Our stated goal is to fit piecewise linear functions to datasets in $\mathcal{R}^2$. In order to do this, we need to mathematically specify exactly what it means for a set of lines to 'fit' a datset. Generally speaking, if we think of a given dataset as a realization of some random variable, we would like our set of lines to represent the center of the density of that random variable. The statement of this idea in mathematical terms is as follows.

Let us assume that we have $h^*$ lines where $1 \leq h^* \leq h_{\max}$, $h_{\max}$ is a known positive integer. For each $i$, $i = 1, \ldots, h^*$, let the equation of the $i^{th}$ line be

$$x\cos\theta_{0i} + y\sin\theta_{0i} = d_{0i}$$

for some $\theta_{0i} \in (0, \pi]$ and $d_{0i} \in \mathcal{R}$. Assume that the $i^{th}$ and $(i+1)^{th}$ lines intersect at the point $(z_{(i+1)1}, z_{(i+1)2})$. We denote the first knot as $(z_{11}, z_{12})$ and the last knot as $(z_{(h^*+1)1}, z_{(h^*+1)2})$.

Let $\epsilon_0 > 0$ and let the set $B_i$ be expressed as

$$B_i = \left\{(x,y) : y \in \left[\frac{d_{0i} - x\cos\theta_{0i}}{\sin\theta_{0i}} - \epsilon_0, \frac{d_{0i} - x\cos\theta_{0i}}{\sin\theta_{0i}} + \epsilon_0\right]; x \in [z_{i1}, z_{(i+1)1}],\ i = 1, \ldots, h^*\right\}$$

We denote $\bigcup_{i=1}^{h^*} B_i$ as $B$. The probability density function on $B$ is denoted by $\alpha : B \to [0, \infty)$, where

$$\alpha(x,y) = \begin{cases} \alpha_i(x,y) & x \in [z_{i1}, z_{(i+1)1}],\ i = 1, \ldots, h^* \\ 0 & \text{otherwise} \end{cases}$$

We define our probability measure $P$ as $P(A) = \int_A \alpha(x,y)\,dx\,dy$ for all Borel $A \subseteq \mathcal{B}(B)$, the Borel $\sigma$-field of $B$.

Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ be independent, identically distributed random vectors with density $\alpha$, i.e., there exists a probability space $(\Omega, \mathcal{A}, Q)$ such that $(X_i, Y_i) : (\Omega, \mathcal{A}, Q) \to (B, \mathcal{B}(B), P),\ i = 1, \ldots, N$, where $Q(C) = P((X_i, Y_i)(C))\ \forall\ C \in \mathcal{A}$.

We also assume that

1. For each $i = 1, \ldots, h^*$,

$$\alpha_i(x,y) = \begin{cases} \alpha_i\left(x, \frac{2(d_{0i} - x\cos\theta_{0i})}{\sin\theta_{0i}} - y\right) & x \in [z_{i1}, z_{(i+1)1}] \\ 0 & x \notin [z_{i1}, z_{(i+1)1}] \end{cases} \tag{8}$$

   (i.e., $\alpha_i$ is symmetric about the line $x\cos\theta_{0i} + y\sin\theta_{0i} = d_{0i}$)

2. $\alpha_i(x,y) > 0\ \forall\ (x,y) \in B_i\ \forall\ i = 1, \ldots, h^*$

3. $\alpha(x,y)$ and $\alpha_i(x,y),\ i = 1, \ldots, h^*$, are continuous.

4. $\int_B \alpha(x,y)\,dx\,dy = 1$

In the above mathematical formulation, the primary assumption is the assumption of symmetry of the underlying density around a piecewise linear function (assumption #1). The other stated assumptions are ordinary properties of a continuous probability density function.

We will fit a piecewise linear function to a given dataset by utilizing a GA to search for the function described above. Hence we need to represent our solution space in a way compatable with the search method of a GA.

## 4.2 Solution Space

We are given a data set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ which is a realization of the random vectors $(X_i, Y_i)$, i.e., $X_i(\omega) = x_i$, $Y_i(\omega) = y_i$ for some $\omega \in \Omega$, $i = 1, \ldots, N$. Define $x_{(1)} = \min\{x_i;\ i = 1, \ldots, N\}$, $x_{(N)} = \max\{x_i;\ i = 1, \ldots, N\}$, $y_{(1)} = \min\{y_i;\ i = 1, \ldots, N\}$, $y_{(N)} = \max\{y_i;\ i = 1, \ldots, N\}$. GAs try to find an optimal solution over a finite solution space. Thus the solution space, i.e., the collection of all $h$-piecewise linear functions where $h \in \mathcal{H}$, $\mathcal{H} = \{1, \ldots, h_{\max}\}$, needs to be discretized. To formulate the problem mathematically, we proceed in the following way.

Let $L_j$ denote the straight line

$$x \cos\theta_j + y \sin\theta_j = d_j$$

where $j$ belongs to an index set. Depending upon the given context, the notation $L_j$ may also refer to the function

$$L_j(x) = \frac{d_j - x\cos\theta_j}{\sin\theta_j} - y$$

Let $L_{j,\cdot,h}$ represent an $h$-piecewise linear function with pieces $(L_{j,1,h}, L_{j,2,h}, \ldots, L_{j,h,h})$ where $L_{j,i,h}$ denotes the $i^{th}$ straight line of the set of $h$ straight lines, $i = 1, \ldots, h$, and each $L_{j,i,h}$ satisfies the following properties:

1. $L_{j,i,h}$ represents the straight line $x\cos\theta_{j,i,h} + y\sin\theta_{j,i,h} = d_{j,i,h}$ where $\theta_{j,i,h}$ $(0 < \theta_{j,i,h} \leq \pi)$ is the polar angle formed when the polar axis is taken as the y-axis and the origin is taken as the intersection point between the y-axis and $L_{j,i,h}$; $d_{j,i,h}$ is the perpendicular distance of the line from the point (0,0).

2. For every $i$, $i = 1, \ldots, h-1$, $L_{j,i,h}$ and $L_{j,(i+1),h}$ intersect and the point of intersection is $(z_{j,(i+1),h,1}, z_{j,(i+1),h,2})$.

3. $z_{j,1,h,1} = x_{(1)}$, $z_{j,(h+1),h,1} = x_{(N)}$, $z_{j,i,h,1} \leq z_{j,(i+1),h,1}$ $\forall\ i = 1, \ldots, h$

4. $L_{j,\cdot,h} \equiv L_{j,i,h}$ if $z_{j,i,h,1} \leq x \leq z_{j,(i+1),h,1}$, $i = 1, \ldots, h$.

The knot locations $((z_{j,1,h,1}, z_{j,1,h,2}), \ldots, (z_{j,(h+1),h,1}, z_{j,(h+1),h,2}))$ of any $L_{j,\cdot,h}$, $1 \leq h \leq h_{\max}$, are not restricted to the datapoints $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$.

For each $h$, $1 \leq h \leq h_{\max}$, let $\mathcal{L}_h$ represent the class of all $h$-piecewise linear functions $L_{j,\cdot,h}$ which satisfy the above properties. Then $\bigcup_{h\in\mathcal{H}} \mathcal{L}_h$ is the collection of functions under consideration.

Note that $\bigcup_{h\in\mathcal{H}} \mathcal{L}_h$ is uncountable. In order to obtain a 'near optimal' function from this space, we need to 'suitably' discretize $\bigcup_{h\in\mathcal{H}} \mathcal{L}_h$, i.e., larger discretizations should lead to finer representations of the solution space. We can achieve this by restricting the values of $\theta$ and $d$. Let $l_a$ be the number of bits used to express $\theta$ and let $l_d$ be the number of bits used to express $d$. We restrict $\theta_{j,i,h}$ to the values

$\{\frac{\pi}{2^{l_a}}, \frac{2\pi}{2^{l_a}}, \ldots, \frac{(2^{l_a}-1)\pi}{2^{l_a}}, \pi\}$. In specifying $d_{j,i,h}$, we utilize the rectangle *rect* formed by the points $(x_{(1)}, y_{(1)})$, $(x_{(N)}, y_{(1)})$, $(x_{(1)}, y_{(N)})$ and $(x_{(N)}, y_{(N)})$. Note that *rect* contains the entire given data set. Let *diag* be the length of the diagonal of *rect* and let $\ell_\theta$ be defined as

$$\ell_\theta = \begin{cases} \mathrm{x}_{(1)} \cos\theta + \mathrm{y}_{(1)} \sin\theta & \text{if } 0 < \theta < \pi/2 \\ \mathrm{x}_{(N)} \cos\theta + \mathrm{y}_{(1)} \sin\theta & \text{if } \pi/2 \leq \theta \leq \pi \end{cases}$$

Then for a given $\theta_{j,i,h}$, $d_{j,i,h}$ may only take values within the set $\{d_{j,i,h} = \ell_{\theta_{j,i,h}} + k_{j,i,h}\delta : k_{j,i,h} \in \{0, 1, \ldots, 2^{l_d} - 1\}, \ \delta = diag/(2^{l_d} - 1)\}$. Note that a line with $d = l_\theta$ intersects *rect* at the point $(x_{(1)}, y_{(1)})$, if $0 < \theta < \pi/2$, or the point $(x_{(N)}, y_{(1)})$, if $\pi/2 \leq \theta < \pi$ (the value $k_{j,i,h}\delta$, $0 \leq k_{j,i,h}\delta \leq diag$, is sometimes referred to as the *offset* value).

For each $h$, $1 \leq h \leq h_{\max}$ let $\mathcal{L}_h^0(\Theta, \mathcal{K})$ denote the finite set of functions in $\mathcal{L}_h$ which satisfy these restrictions, where $\theta$ has $\Theta$ possible values and $k$ has $\mathcal{K}$ possible values (*note that* $\Theta = 2^{l_a}$ *and* $\mathcal{K} = 2^{l_d}$). Then $\mathcal{L}_h^0(\Theta, \mathcal{K})$ may be expressed as

$$\begin{aligned} \mathcal{L}_h^0(\Theta, \mathcal{K}) = \{L_{j,\cdot,h} \in \mathcal{L}_h : L_{j,i,h} &\text{ is of the form} \\ x\cos\theta_{j,i,h} + y\sin\theta_{j,i,h} &= \ell_{\theta_{j,i,h}} + k_{j,i,h}\delta \ \ \forall \ \ i = 1, \ldots, h, \\ \theta_{j,i,h} \in \{\tfrac{\pi}{2^{l_a}}, \tfrac{2\pi}{2^{l_a}}, \ldots, \tfrac{(2^{l_a}-1)\pi}{2^{l_a}}, \pi\}, \ k_{j,i,h} &\in \{0, 1, \ldots, 2^{l_d} - 1\}, \text{ and } \delta = diag/(2^{l_d} - 1)\}. \end{aligned}$$

Hence $\{\mathcal{L}_h^0(\Theta, \mathcal{K}) : \mathrm{l_a} = 1, 2, \ldots\}$ and $\{\mathcal{L}_h^0(\Theta, \mathcal{K}) : \mathrm{l_d} = 1, 2, \ldots\}$ represent increasing sequences of nested sets. For sake of clarity, we will henceforth specify $L_{j,\cdot,h}$ as $L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})$ and $L_{j,\cdot,h}(x)$ as $L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})(x)$. Note that $\boldsymbol{\theta}_{j,\cdot,h}$ and $\boldsymbol{k}_{j,\cdot,h}$ represent the vectors $(\theta_{j,1,h}, \ldots, \theta_{j,h,h})$ and $(k_{j,1,h}, \ldots, k_{j,h,h})$.

For $L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}) \in \mathcal{L}_h^0(\Theta, \mathcal{K})$ we define

$$E_{\epsilon,a,b}(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})) = \{(x, y) : y \in (L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})(x) - \epsilon, L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})(x) + \epsilon), \ x \in [a, b]\}$$

for $a < b$, $\epsilon > 0$, and let $E_\epsilon(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})) = \bigcup_b \bigcup_{a<b} E_{\epsilon,a,b}(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}))$.

If we recall the piecewise linear function which lies at the center of the density of $(x, y)$ and we denote it as $f$, i.e.,

$$f(x) = \begin{cases} \frac{d_{0i} - x\cos\theta_{0i}}{\sin\theta_{0i}} & x \in [\mathrm{z}_{i1}, \mathrm{z}_{(i+1)1}], \ \text{ for } i = 1, \ldots, h^* \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

then the support $B$ of the density is equivalent to $E_{\epsilon_0, \mathrm{z}_{11}, \mathrm{z}_{(h^*+1)1}}(f)$.

Corresponding to $E_\epsilon(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}))$ we define the set of piecewise linear functions

$$\mathcal{L}_h^{(\epsilon)} = \{L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}) \in \mathcal{L}_h : P(E_\epsilon(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}))) \geq 0.95\}$$

where $P(A) = \int_A \alpha(x, y)\, dx\, dy$ for $A$ Borel, $A \subseteq \mathcal{B}(B)$. $\mathcal{L}_h^{(\epsilon)}$ represents those functions $L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}) \in \mathcal{L}_h$ for which the probability of the region $\{(x, y) : y \in (L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})(x) - \epsilon, L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})(x) + \epsilon), \ x \in [z_{11}, z_{(h^*+1)1}]\}$ is greater than 0.95. Just as $\{\mathcal{L}_h^0(\Theta, \mathcal{K}) : \mathrm{l_a} = 1, 2, \ldots\}$ and $\{\mathcal{L}_h^0(\Theta, \mathcal{K}) : \mathrm{l_d} = 1, 2, \ldots\}$ represent increasing sequences of sets, we have $\mathcal{L}_h^{(\epsilon_1)} \subseteq \mathcal{L}_h^{(\epsilon_2)}$ if $\epsilon_1 \leq \epsilon_2$. These definitions involving $E_\epsilon(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}))$ will be used to define our optimization criterion.

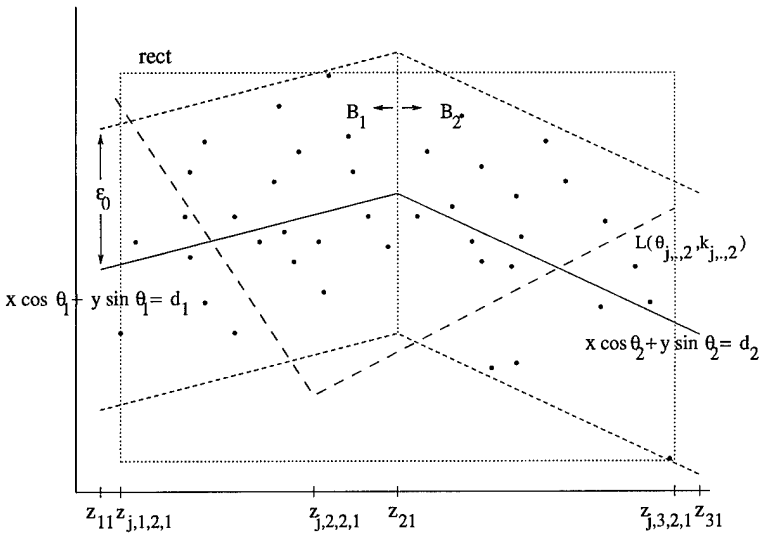



Figure 1: Support $B = B_1 \bigcup B_2$ with center lines; Example function from $\mathcal{L}_2^0(\Theta, \mathcal{K})$

Figure 1 shows an example dataset where the support $B = B_1 \bigcup B_2$ of the density $\alpha$ is centered around a 2-piece linear function. A function from $\mathcal{L}_2^0$ and *rect* are also shown.

This completes an outline of the mathematical context of our problem. We now look at the function to be optimized.

## 4.3 Optimization Criterion

The function which is chosen to represent a given dataset is often that which minimizes the sum of the squared error (i.e., the least squares function). However, a criterion based on least squares often yields a solution that is not **robust** - outliers in the dataset can pull the least squares function away from most of the data points and hence away from $f$ [26, 28]. For this reason we will not to use least squares as our optimization criterion. Instead, we note that our concept of a 'fitted' function is one which represents the center of a symmetric density function. Given a dataset $D$, if $L(\boldsymbol{\theta}_{j,\cdot,h'}, \boldsymbol{k}_{j,\cdot,h'})$ represents our 'fitted' function then the majority of the data points in $D$ should fall within the region $\{(x,y) : y \in (L(\boldsymbol{\theta}_{j,\cdot,h'}, \boldsymbol{k}_{j,\cdot,h'})(x) - \epsilon, L(\boldsymbol{\theta}_{j,\cdot,h'}, \boldsymbol{k}_{j,\cdot,h'})(x) + \epsilon),\ x \in [z_{j,1,h',1}, z_{j,(h'+1),h',1}]\}$ for some 'small' $\epsilon > 0$. This observation is the basis of our optimization criterion.

Let $\psi_{j,\cdot,h}$ denote an $h$-piecewise linear function where each piece $\psi_{j,i,h},\ i = 1, \ldots, h$, satisfies the properties listed for $L_{j,i,h}$. For any $\epsilon > 0$, define

$$\mathcal{I}_{\epsilon,\psi_{j,\cdot,h}}(x,y) = \begin{cases} 1 & \mid y - \psi_{j,\cdot,h}(x) \mid < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

The proposed function to be optimized (i.e., fitness function) may then be stated as

$$FF_{\epsilon,N}(\psi_{j,\cdot,h}) = \sum_{i=1}^{N} \mathcal{I}_{\epsilon,\psi_{j,\cdot,h}}(\mathrm{x}_i, \mathrm{y}_i) + (1 - \frac{h}{h_{\max}}) \qquad (10)$$

Note the similarity in form to the fitness function of the VLGA (Eqn. 7). Our solution space $\mathcal{L}^0(\Theta, \mathcal{K}) = \bigcup_{h \in \mathcal{H}} \mathcal{L}_h^0(\Theta, \mathcal{K})$ for some $\Theta$ and $\mathcal{K}$ represents the collection of functions under consideration. By the continuity of $\alpha(x, y)$, with $\mathcal{L}^{(\epsilon)} = \bigcup_{h \in \mathcal{H}} \mathcal{L}_h^{(\epsilon)}$, we know there exists some $\epsilon^* : \mathcal{L}^{(\epsilon^*)} \neq \emptyset$ and $\mathcal{L}^{(\epsilon)} = \emptyset \; \forall \; \epsilon < \epsilon^*$ (i.e., there exists some function $L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}) \in \mathcal{L}_h$ for some value $h \in \mathcal{H}$ for which $P(E_\epsilon(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}))) \geq 0.95$ when $\epsilon \geq \epsilon^*$ but no such function exists when $\epsilon < \epsilon^*$). To balance accuracy and robustness, we specify that at least 95% of the data points in $D$ (not necessarily 100%) fall within $\epsilon^*$ of the optimal function - our solution belongs to $\mathcal{L}^{(\epsilon^*)}$. This is equivalent to setting $crit = 5\%$ (see Section 3.4). The choice of 95% is **heuristic** - it can be altered depending upon the characteristics of the dataset (e.g., the percentage of outliers) and the desired precision. Hence our goal is to use genetic algorithms to find an optimal $h'$-piecewise linear function $L(\boldsymbol{\theta}_{j^*,\cdot,h'}, \boldsymbol{k}_{j^*,\cdot,h'}) \in \mathcal{L}^0(\Theta, \mathcal{K}) \bigcap \mathcal{L}^{(\epsilon^*)}$ such that

$$FF_{\epsilon^*,N}(L(\boldsymbol{\theta}_{j^*,\cdot,h'}, \boldsymbol{k}_{j^*,\cdot,h'})) = \max_{L(\theta_{j,\cdot,h}, k_{j,\cdot,h}) \in \mathcal{L}^0(\Theta,\mathcal{K})} FF_{\epsilon^*,N}(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h})) \qquad (11)$$

where $h' = \min\{h \in \mathcal{H} : \mathcal{L}_h^{(\epsilon^*)} \neq \emptyset\}$.

We observe that if we require that the entire dataset $D$ fall within $\epsilon^*$ of the optimal function then, for $\Theta$ and $\mathcal{K}$ sufficiently large, as the number of iterations and $N \to \infty$ the optimal function should correspond to the least squares function. This will be discussed in more detail in the following section.

For the sake of clarity, a function will be referred to as **optimal** if (1) (1 - *crit*)% of the data points fall within $\epsilon^*$ of the function and (2) of all functions for which this is true, the function contains the least number of lines. Note that this is *different* from an optimal solution. An optimal solution is the optimal string to which an elitist GA converges and it represents the string which maximizes the given fitness function. A function will be referred to as **properly** fitted (or **proper**) if at least (1-*crit*)% of the data points fall within $\epsilon$ of the function for a specified $\epsilon > 0$ (hence optimal functions are also proper). Note that $\mathcal{L}_h^{(\epsilon)}$ represents the set of proper functions in $\mathcal{L}_h^0$. Unless otherwise specified, *crit* will be taken as 0.05.

Our goal can be attained if and only if

1. Our search space $\mathcal{L}^0(\Theta, \mathcal{K})$ contains an optimal function as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.
2. An optimal solution represents an optimal function, i.e., a function such that at least 95% of the data points fall within $\epsilon^*$ of the function and of all functions for which this is true, the function contains the least number of lines.
3. The proposed algorithm converges to an optimal solution.

Statement #3 will be discussed in Section 4.5.2. Statements #1 and #2 will be justified in detail for the case $h_{\max} = 1$ and these results will be extended to the case of $h_{\max} > 1$, $h^*$ unknown, $h^* \in \mathcal{H}$.

## 4.4 Case $h_{\max} = 1$

### 4.4.1 Optimal Function in Search Space

In the case where $h_{\max} = 1$, $h^*$ and $h'$ are both known to be 1.

Recall

$$\mathcal{L}_1^0(\Theta, \mathcal{K}) = \{L(\theta_{j,1,1}, k_{j,1,1}) \in \mathcal{L}_1 : \ L(\theta_{j,1,1}, k_{j,1,1}) \text{ is of the form } x\cos\theta_{j,1,1} + y\sin\theta_{j,1,1} = \ell_{\theta_{j,1,1}} + k_{j,1,1}\delta, \text{ where } \theta_{j,1,1} \text{ is one of } \Theta \text{ values, } k_{j,1,1} \text{ is one of } \mathcal{K} \text{ values}\}$$

Let

$$\mathcal{P}_1 = \{L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{L}_1 : \ \exists\ (x_r, y_r) \in \textit{rect} \text{ satisfying } \ L(\theta_{m,1,1}, k_{m,1,1}), \ 0 < \theta_{m,1,1} \leq \pi, \ k_{m,1,1} \in \mathcal{R}\}.$$

$\mathcal{P}_1$ represents the set of all lines in $\mathcal{L}_1$ which intersect *rect*.

We will justify the following statements:

I. For any fixed $\epsilon \geq \epsilon^*$ our class $\mathcal{L}_1^0(\Theta, \mathcal{K})$ will contain a proper line (i.e., a line in $\mathcal{L}_1^{(\epsilon)}$) as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.

II. For any fixed $\epsilon \geq \epsilon^*$ the set of proper lines in $\mathcal{L}_1^0(\Theta, \mathcal{K})$ increases to the set of proper lines in $\mathcal{P}_1$ as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.

Additionally, we will show

III. For $0 < p \leq 1$ let $\mathcal{A}^{(\epsilon)}(p) = \{L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{P}_1 : P(E_\epsilon(L(\theta_{m,1,1}, k_{m,1,1}))) \geq p\}$, and let $\epsilon_p^* : \mathcal{A}^{(\epsilon_p^*)}(p) \neq \emptyset$ and $\mathcal{A}^{(\epsilon)}(p) = \emptyset \ \ \forall \ \ \epsilon < \epsilon_p^*$. Then for $p = 1.0$ and $N \to \infty$, $\mathcal{A}^{(\epsilon_p^*)}(p)$ converges to a unique optimal function $L(\theta_{j^*,1,1}, k_{j^*,1,1}) \in \mathcal{P}_1$

As the discretization of $\mathcal{L}_1$ becomes finer (i.e., as $\Theta \to \infty$ and $\mathcal{K} \to \infty$), Statement I says that for any $\epsilon \geq \epsilon^*$ our solution space will contain **a** proper function, hence when $\epsilon = \epsilon^*$ our solution space will contain **an** optimal function. Similarly, Statement II indicates that for any $\epsilon \geq \epsilon^*$ our solution space will contain **all** proper functions (hence all optimal functions) which intersect *rect*. Statement III says that as our critical level increases to 1.0 (i.e., as we require that a larger percentage $p$ of the data points fall within $\epsilon_p^*$ of the 'fitted' function) and $N \to \infty$, the set of optimal functions intersecting *rect* decreases to a single function. This unique function is the least squares function.

Note that we have restricted ourselves to considering only those functions which intersect *rect* so our solution space is actually $\mathcal{L}_1^0(\Theta, \mathcal{K})$ restricted to those lines which intersect *rect* (and similarly for $\mathcal{L}_1^{(\epsilon)}$). This is validated by the following theorem:

**Theorem 4.1** For any $\epsilon \geq \epsilon^*$ and for $\Theta$ and $\mathcal{K}$ sufficiently large, there exists an proper line $L(\theta_{j^{**},1,1}, k_{j^{**},1,1}) \in \mathcal{L}_1$ such that

$$\{(x,y) : y = L(\theta_{j^{**},1,1}, k_{j^{**},1,1})(x),\ x \in [z_{11}, z_{21}]\} \bigcap rect \neq \emptyset$$

A proof of Theorem 4.1 and a graphical representation of the proof are provided in the appendix.

From Theorem 4.1 we know that if a proper function exists, then there exists a proper function which intersects *rect*. It follows that without loss of generality, given $\epsilon \geq \epsilon^*$ we can restrict ourselves to $\Theta$ and $\mathcal{K}$ sufficiently large and only those proper lines which intersect *rect*, i.e., those proper lines which belong to $\mathcal{P}_1$.

We now justify Statements I, II, and III with the help of the following propositions and theorems, the proofs of which can be found in the appendix.

For simplicity, let $\rho = \ell_\theta + k\delta$ for given $\theta$ and $k$.

Statement I will be justified if we can show that for $\Theta$ and $\mathcal{K}$ large and any $\epsilon \geq \epsilon^*$, given any proper line in $\mathcal{P}_1$ we can find a line in $\mathcal{L}_1^0(\Theta, \mathcal{K})$ which is arbitrarily close to it. This is implied by Proposition 4.1 which states that given any line in $\mathcal{P}_1$ we can find a $\Theta$ and $\mathcal{K}$ such that there exists a line in $\mathcal{L}_1^0(\Theta, \mathcal{K})$ which is arbitrarily close to the given line.

**Proposition 4.1** Let $L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{P}_1$. Let $\xi > 0$. Then $\exists\ (\Theta_\xi, \mathcal{K}_\xi)$ such that $\forall\ \Theta > \Theta_\xi$ and $\mathcal{K} > \mathcal{K}_\xi$, $\exists\ L(\theta, k)$ for which

1. $L(\theta, k) \in \mathcal{L}_1^0(\Theta, \mathcal{K})$

   and

2. $\mid \theta - \theta_{m,1,1} \mid < \xi/2$ and $\mid \rho - \rho_{m,1,1} \mid < \xi/2$.

Now, given $\xi > 0$, we would like there to exist a $\Theta_\xi$ and $\mathcal{K}_\xi$ such that for **any** $\Theta > \Theta_\xi$, $\mathcal{K} > \mathcal{K}_\xi$, given **any** proper line in $\mathcal{P}_1$, $\mathcal{L}_1^0(\Theta, \mathcal{K})$ will contain a line which is arbitrarily close to the given proper line. We know such a $(\Theta_\xi, \mathcal{K}_\xi)$ exists by the following theorem.

**Theorem 4.2** For each $\xi > 0$, $\exists\ (\Theta_\xi, \mathcal{K}_\xi)$ : for all $\Theta > \Theta_\xi$ and for all $\mathcal{K} > \mathcal{K}_\xi$, given any $L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{P}_1$, $\exists\ L(\theta, k)$ :

1. $L(\theta, k) \in \mathcal{L}_1^0(\Theta, \mathcal{K})$

   and

2. $| \theta - \theta_{m,1,1} | < \xi/2$ and $| \rho - \rho_{m,1,1} | < \xi/2$.

Thus given $\Theta$ and $\mathcal{K}$ sufficiently large and any $\epsilon \geq \epsilon^*$ we can get a line which is arbitrarily close to a proper (or optimal) line. Hence Statement I is justified.

For the purpose of justifying Statement II, let $\{\Theta_i : \Theta_i \leq \Theta_{i+1},\ i = 1, 2, \dots\}$ and $\{\mathcal{K}_i : \mathcal{K}_i \leq \mathcal{K}_{i+1},\ i = 1, 2, \dots\}$ represent the possible values of $\Theta$ and $\mathcal{K}$. For any $\epsilon > 0$ we define

$$\mathcal{A}_{\epsilon i}(p) = \{L(\theta_{j_i^*,1,1}, k_{j_i^*,1,1}) \in \mathcal{L}_1^0(\Theta_i, \mathcal{K}_i) : P(E_{\epsilon, L(\theta_{j_i^*,1,1}, k_{j_i^*,1,1})}) \geq p\}$$

and

$$\mathcal{A}_{\epsilon}(p) = \{L(\theta_{m^*,1,1}, k_{m^*,1,1}) \in \mathcal{P}_1 : P(E_{\epsilon}(L(\theta_{m^*,1,1}, k_{m^*,1,1}))) \geq p\}$$

and let $\mathcal{A}_{\epsilon i} = \mathcal{A}_{\epsilon i}(0.95)$ and $\mathcal{A}_{\epsilon} = \mathcal{A}_{\epsilon}(0.95)$. $\mathcal{A}_{\epsilon i}$ represents the set of all proper lines which belong to $\mathcal{L}_1^0(\Theta_i, \mathcal{K}_i)$ while $\mathcal{A}_{\epsilon}$ represents the set of all proper lines which belong to $\mathcal{P}_1$. We would like $\mathcal{A}_{\epsilon i} \to \mathcal{A}_{\epsilon}$ as $i \to \infty$. However, we first need that if $\Theta \to \infty$ and $\mathcal{K} \to \infty$ then any proper line which is the limit of any sequence of functions in $\mathcal{P}_1$ is contained in $\mathcal{P}_1$ (and hence belong to our search space). This is, in fact, true, as stated in Proposition 4.2.

**Proposition 4.2** For each $i = 1, 2, \dots$, let $L(\theta_{n_i,1,1}, k_{n_i,1,1}) \in \mathcal{L}_1^0(\Theta_i, \mathcal{K}_i)$ : $\theta_{n_i,1,1} \to \theta_{\lim}$ and $k_{n_i,1,1} \to k_{\lim}$ for some $\theta_{\lim}$, $0 < \theta_{\lim} \leq \pi$, and $k_{\lim}$, $0 \leq k_{\lim} < \infty$, as $i \to \infty$. Let

$$\mathcal{T}_1 = \{L(\theta_{\lim}, k_{\lim}) : \exists \text{ a sequence } \{L(\theta_{n_i,1,1}, k_{n_i,1,1})\}_{i=1}^{\infty},\ L(\theta_{n_i,1,1}, k_{n_i,1,1}) \in \mathcal{L}_1^0(\Theta_i, \mathcal{K}_i) \text{ such that } \theta_{n_i,1,1} \to \theta_{\lim} \text{ and } k_{n_i,1,1} \to k_{\lim}\}$$

Then given any $\epsilon \geq \epsilon^*$ the proper lines in $\mathcal{P}_1$ are the proper lines in $\mathcal{T}_1$.

This implies that $\mathcal{P}_1$ contains any proper (or optimal) function which is a limit of a sequence of functions in $\mathcal{P}_1$. We may now justify Statement II.

Note that the fitness function $FF_{\epsilon,N} : (0, \pi] \times [-M, M] \to [0, \infty)$ is continuous where for any line in $\mathcal{T}_1$, $\theta$ belongs to $(0, \pi]$ and the distance of the line from the origin is less than $M$. With $FF_{\epsilon,N}$ continuous and bounded, we state Theorem 4.3.

**Theorem 4.3** Let $\mathcal{A}_{\epsilon i}$, $i = 1, 2, \dots$, and $\mathcal{A}_{\epsilon}$ be as defined above. Then $\mathcal{A}_{\epsilon i} \to \mathcal{A}_{\epsilon}$ as $i \to \infty$.

Hence the proper (or optimal) lines which intersect *rect* are in the search space as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.

The above theorem holds for any $\mathcal{A}_{\epsilon}(p)$, $0.95 \leq p \leq 1.0$, and for any $\epsilon \in \{\epsilon_p^*\}_{p=0.95}^{1}$, where $\epsilon_p^* : \mathcal{A}_{\epsilon_p^*}(p) \neq \emptyset$ and $\mathcal{A}_{\epsilon}(p) = \emptyset\ \forall\ \epsilon < \epsilon_p^*$ (i.e., if we require that 96% of the data

points in a given dataset fall within $\epsilon$ of our fitted function, then $\epsilon^*_{0.96}$ is the smallest $\epsilon > 0$ for which such a function exists). With this in mind, we conclude with a theorem which justifies Statement III.

**Theorem 4.4** Let $\mathcal{A}_\epsilon(p)$ and $\epsilon^*_p$ be as defined, $0.95 \leq p \leq 1.0$. Then $\mathcal{A}_{\epsilon^*_{1.0}}(1.0)$ converges to a single optimal line $L(\theta_{j^*,1,1}, k_{j^*,1,1})$ as $N \to \infty$.

A graphical representation of this theorem is provided in the Appendix.

Suppose we require that all of the data points fall within $\epsilon^*_{1.0}$ of the 'fitted' function. Then for $\Theta$ and $\mathcal{K}$ large, as $N \to \infty$ our optimal function will converge to the least squares function. This implies that the proposed algorithm can be used to fit non-robust as well as more robust optimal functions.

By justifying the above statements we have shown that our search space contains an optimal function. It remains to be shown that the optimal solution of the proposed GA is an optimal function for $h_{\max} = 1$.

### 4.4.2 Optimal Solution

For any $\epsilon \geq \epsilon^*$ we know from the above section that we may choose $\Theta$ and $\mathcal{K}$ sufficently large so that a properly fitted function is contained in the search space $\mathcal{L}^0_1(\Theta, \mathcal{K})$.

As defined previously, let

$$\mathcal{I}_{\epsilon,L(\theta_{j,1,1},k_{j,1,1})}((x,y)) = \begin{cases} 1 & \mid y - L(\theta_{j,1,1}, k_{j,1,1})(x) \mid < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

Since $h$ takes only one value, we may drop the term $(1 - \frac{h}{h_{\max}})$ from Eqn. 10 and allow

$$FF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j1}, k_{j1})) = \textstyle\sum_{i=1}^{N} \mathcal{I}_{\epsilon,L(\theta_{j1},k_{j1})}((\mathrm{x}_i, \mathrm{y}_i))$$

We write $FF_{\epsilon,N,\Theta,\mathcal{K}}$ as opposed to $FF_{\epsilon,N}$ to emphasize the dependence of the optimal solution on $\Theta$ and $\mathcal{K}$.

Define

$$AFF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j,1,1}, k_{j,1,1})) = \tfrac{1}{N} FF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j,1,1}, k_{j,1,1}))$$

Note that

$$\lim_{N\to\infty} AFF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j,1,1}, k_{j,1,1})) = P(\textstyle\bigcup_b \bigcup_{a<b} E_{\epsilon,a,b}(L(\theta_{j,1,1}, k_{j,1,1}))) = P(E_\epsilon(L(\theta_{j,1,1}, k_{j,1,1})))$$

Given the convergence of the elitist GA, we know that by including elitism in the proposed algorithm we can ensure that it will find an optimal solution $L(\theta_{j,1,1}, k_{j,1,1})$ for a given $\mathcal{L}^0(\Theta, \mathcal{K})$, $\epsilon$, $N$, and a sufficiently large number of iterations. So let $L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon,N}$ be such that

$$FF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon,N}) = \max_{L(\theta_{j,1,1},k_{j,1,1})\in\mathcal{L}_1^0(\Theta,\mathcal{K})} FF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j,1,1}, k_{j,1,1}))$$

where the dependence of an optimal solution on $\epsilon$ and $N$ has been made explicit. $L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon,N}$ also maximizes $AFF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j,1,1}, k_{j,1,1}))$. To determine whether the algorithm to be proposed converges to an optimal function, we examine $\lim_{N\to\infty} AFF_{\epsilon,N,\Theta,\mathcal{K}}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon,N})$ for $\epsilon = \epsilon_{2N}$ where $AFF_{\epsilon_{2N},N,\Theta,\mathcal{K}}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon_{2N},N}) \geq 0.95$. If $\lim_{N\to\infty} AFF_{\epsilon_{2N},N,\Theta,\mathcal{K}}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon_{2N},N})$ is at least 0.95, then the proposed algorithm does indeed converge to an optimal function (i.e., the optimal solution represents an optimal function). Theorem 4.5 states that this is indeed true.

**Theorem 4.5** Let $\epsilon_{2N}$ be as defined above. Then for appropriate $\Theta$ and $\mathcal{K}$

$$\liminf_{N\to\infty} AFF_{\epsilon_{2N},N}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon_{2N},N}) \geq 0.95$$

The proof of Theorem 4.5 is presented in the Appendix.

Hence we have established that for $h_{\max} = 1$,

1. Our search space $\mathcal{L}^0(\Theta, \mathcal{K})$ contains an optimal function as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.
2. The optimal solution represents an optimal function, i.e., a function such that at least 95% of the data points fall within $\epsilon^*$ of the function and of all functions for which this is true, the function contains the least number of lines.

### 4.4.3 Remarks

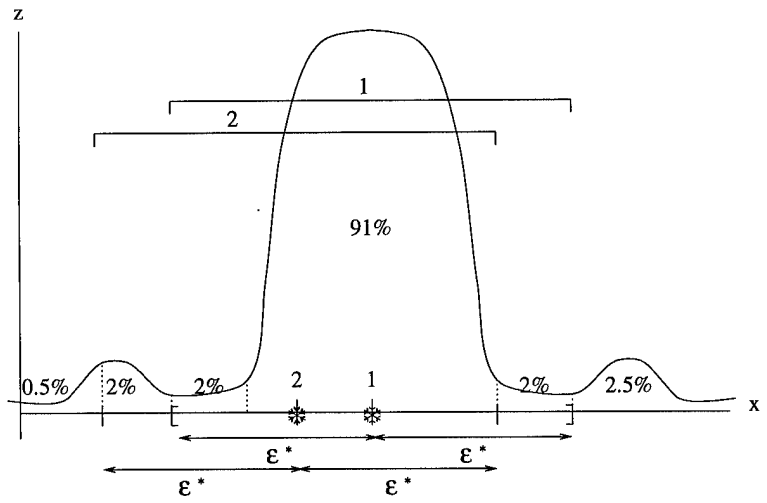
1. For large $\Theta_{i_0}$ and $\mathcal{K}_{i_0}$, $\mid \theta_i - \theta_{i-1} \mid$ and $\mid k_i - k_{i-1} \mid$ are both small, so an optimal line $L(\theta_{j_i^*,1,1}, k_{j_i^*,1,1})$ in $\mathcal{L}^0(\Theta_{i_0}, \mathcal{K}_{i_0})$ will be close to an optimal line $L(\theta_{m^*,1,1}, k_{m^*,1,1})$ in $\mathcal{P}_1$ in terms of Euclidean distance.
2. A search procedure based on the above mathematical formulation may be designed so that $\Theta$, $\mathcal{K}$, and $\epsilon$ are adjusted adaptively, i.e., in a way that is dependent upon the algorithm's result. For example, we may start with initial choices $\Theta_{i_0}$, $\mathcal{K}_{i_0}$, and $\epsilon_{i_0}$, and run the algorithm for a finite number of iterations. If we reach several proper results, we may reduce $\epsilon_{i_0}$; if we reach a single proper result, we may increase $\Theta$ and $\mathcal{K}$. We then repeat this process until a 'satisfactory' solution has been achieved, i.e., a 'fitted' function which reflects the center of the probability density function of the observed random variables with an acceptable level of precision. Note that if $\Theta$ and (or) $\mathcal{K}$ are (is) small or $\epsilon$ is large, it is possible for the algorithm's result to be close to an optimum in terms of probability but not in terms of Euclidean distance. Since appropriate values for $\Theta$, $\mathcal{K}$, and $\epsilon$ are unknown apriori, we need to implement an **adaptive** procedure.

3. Recall that our requirement that 95% of the data points be within $\epsilon^*$ of the optimal solution is **heuristic**. Depending upon the particular dataset at hand and the desired accuracy of the solution, we may alter this criterion to better suit our needs. For example, we may choose our *critical level* to be 97% if our dataset has no outliers or 90% if our dataset has several outliers.

4. In Statement III and its corresponding proof (Theorem 4.4) we justified that with a critical level of 1.0 and $\epsilon = \epsilon^*$ the algorithm converges to a unique optimum as $N \to \infty$ for a sufficiently large number of iterations. Note that this unique optimum corresponds to the least squares solution (which is nothing but the line represented by Eqn. 9 for $h^* = 1$). Hence our method can be used to fit both robust (using an $\epsilon$ criterion) and non-robust (using a least squares criterion) solutions.

5. It is possible for our optimization problem to have more than one optimal solution. For example, consider Figure 2 which shows a cross-section of the density of the observed random vector (X, Y). Suppose we have two parallel lines lying in the $x-y$ plane, one located at the center of the interval marked with a number 1 (crossing the $x$-axis at the star marked with a number 1) and one located at the center of the interval marked with a number 2 (crossing the $x$-axis at the star marked with a number 2). The percent values on the graph indicate the percentage of the data points with an $x$ value lying within the corresponding marked interval. Using these percentages we see that 95% of the data points fall within $\epsilon$ of line #1 and 95% of the data points fall within $\epsilon$ of line #2. Hence both of these lines would satisfy our optimization criteria (Eqn. 11).

Figure 2: The existence of two optimal lines for a given data set




6. We have assumed that the support of $\alpha_1(x, y)$, $B$, is rectangular in shape. However, $B$ may have curved symmetric boundaries as opposed to straight lines. For example, let $f(x)$ be as defined in Eqn. 9, i.e.,

$$f(x) = \begin{cases} \frac{d_{01} - x\cos\theta_{01}}{\sin\theta_{01}} & x \in [z_{11}, z_{21}] \\ 0 & \text{otherwise} \end{cases}$$

Then we could have $B$ as shown in Figure 3. All of the above results except Theorem 4.4 hold for such a support $B$ as long as the symmetricity of $\alpha_1(x, y)$ is maintained. The symmetricity is essential due to the nature of $FF_{\epsilon,N}$. For Theorem 4.4 to hold we also require that $\gamma_1 \neq \gamma_2$ (so that the center portion of $f$ has positive width).

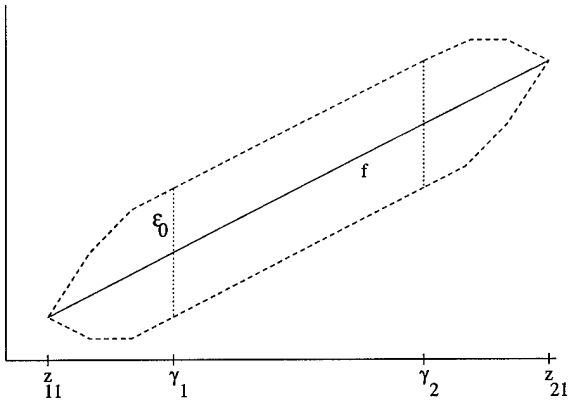



Figure 3: A dataset whose density has a support $B$ with curved boundaries

## 4.5 Case $h_{\max} > 1$

Having justified Statements #1 and #2 for the case $h_{\max} = 1$, we now consider the case of $h_{\max} > 1$, $h^*$ unknown, and $h^* \in \mathcal{H}$. We initially extend the results of Section 4.4 to the case of $h_{\max} > 1$, $h^*$ known, and $h^* \in \mathcal{H}$.

### 4.5.1 Optimal Function in Search Space

Suppose $h = h^*$, $h^* > 1$, $h^* \in \mathcal{H}$ where $h^*$ is known.

Recall that our search space is

$$\mathcal{L}^0_{h^*}(\Theta, \mathcal{K}) = \{L_{j,\cdot,h^*} \in \mathcal{L}_{h^*} : L_{j,i,h^*} \text{ is of the form}$$
$$x \cos\theta_{j,i,h^*} + y \sin\theta_{j,i,h^*} = \ell_{\theta_{j,i,h^*}} + k_{j,i,h^*}\delta \;\; \forall \;\; i = 1, \ldots, h^*,$$
$$\theta_{j,i,h^*} \in \{\tfrac{\pi}{2^{l_a}}, \tfrac{2\pi}{2^{l_a}}, \ldots, \tfrac{(2^{l_a}-1)\pi}{2^{l_a}}, \pi\},\ k_{j,i,h^*} \in \{0, 1, \ldots, 2^{l_d}-1\}, \text{ and } \delta = diag/(2^{l_d}-1)\}.$$

where each $L(\theta_{j,i,h^*}, k_{j,i,h^*})$, $i = 1, \ldots, h^*$, satisfies the properties listed in Section 4.2 for $L_{j,i,h}$.

With respect to optimization we can approach each $L_{j,i,h^*}$ as we did the $h_{\max} = 1$ case with $\alpha_i(x, y)$ as $\alpha_1(x, y)$ and $[z_{i1}, z_{(i+1)1}]$ as $[z_{11}, z_{12}]$. We then recognize $\mathcal{L}^0_{h^*}(\Theta, \mathcal{K})$ as simply $\bigcup_{i=1}^{h^*} \bigcup_j \mathcal{L}^0_{h^*,i,j}(\Theta, \mathcal{K})$ where

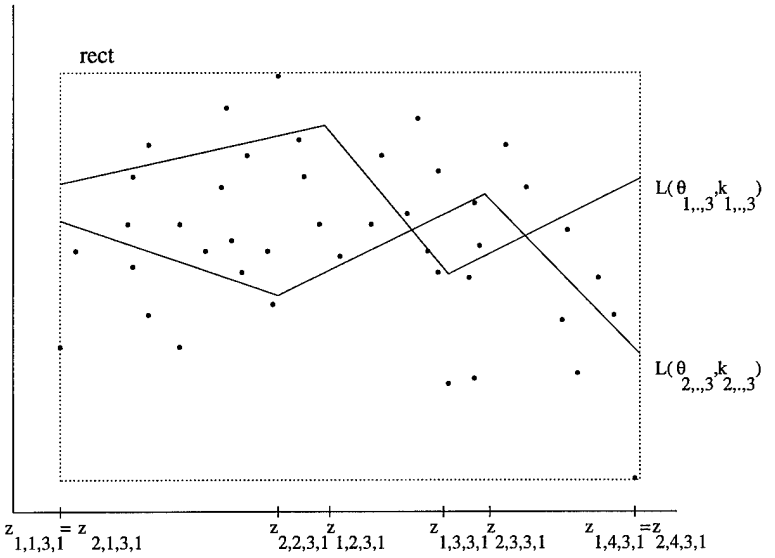



Figure 4: Example functions from $\mathcal{L}_3^0$

- $\mathcal{L}^0_{h^*,i,j}(\Theta, \mathcal{K})$ is $\mathcal{L}^0_1(\Theta, \mathcal{K})$ restricted to $[z_{j,i,h^*,1}, z_{j,i+1,h^*,1}]$

  and

- the union is taken over all possible knot locations and over all possible pieces (any knot/piece combinations which contain pieces that do not intersect are removed).

By the results of Section 4.4.1, for any $i$, $j$ we can get arbitrarily close to any line over $[z_{j,i,h^*,1}, z_{j,i+1,h^*,1}]$ intersecting *rect*, hence we can get arbitrarily close to any piecewise function with knot locations whose $x$ coordinates are $[z_{j,1,h^*,1}, \ldots, z_{j,(h^*+1),h^*,1}]$ and whose pieces intersect *rect*. By taking the union over all possible knot locations with $x$ coordinates $[z_{j,1,h^*,1}, \ldots, z_{j,(h^*+1),h^*,1}]$ and over all $j$, we see that an optimal $h^*$-piecewise function is contained in $\mathcal{L}^0_{h^*}(\Theta, \mathcal{K})$ as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.

By defining our search space as $\mathcal{L}^0(\Theta, \mathcal{K}) = \bigcup_{h \in \mathcal{H}} \mathcal{L}^0_h(\Theta, \mathcal{K})$ (see Section 4.3) we can guarantee, using the same reasoning as in Section 4.4.1, that an optimal $h^*$-piecewise function, $h^*$ unknown, $h^* \in \mathcal{H}$, will be contained in the search space as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.

To show that the optimal solution is a properly fitted function, we first consider the case of $h^*$ known.

### 4.5.2 Optimal Solution

If $h^*$ is known, $h^* \in \mathcal{H}$, then for appropriate $\Theta$ and $\mathcal{K}$ the proposed algorithm is searching for a function $L(\boldsymbol{\theta}_{j^*,\cdot,h^*}, \boldsymbol{k}_{j^*,\cdot,h^*}) \in \mathcal{L}^0_{h^*}(\Theta, \mathcal{K}) \bigcap \mathcal{L}^{(\epsilon^*)}_{h^*}$ which maximizes

$$FF_{\epsilon^*,N,\Theta,\mathcal{K}}(L(\boldsymbol{\theta}_{j,\cdot,h^*}, \boldsymbol{k}_{j,\cdot,h^*})) = \sum_{i=1}^{N} \mathcal{I}_{\epsilon^*,L(\theta_{j,\cdot,h^*},k_{j,\cdot,h^*})}(\mathrm{x}_i, \mathrm{y}_i) + (1 - \frac{h^*}{h_{\max}}) \qquad (12)$$

over all $L(\boldsymbol{\theta}_{j,\cdot,h^*}, \boldsymbol{k}_{j,\cdot,h^*}) \in \mathcal{L}^0_{h^*}(\Theta, \mathcal{K})$. As was done in Section 4.4.2, we may drop the term $(1 - \frac{h^*}{h_{\max}})$. The convergence of the elitist GA combined with the continuity of $FF_{\epsilon,N}$ ensures that Theorem 4.5 holds, i.e.,

$$\liminf_{N\to\infty} AFF_{\epsilon_{2N},N}(L(\boldsymbol{\theta}_{j^*,\cdot,h^*}, \boldsymbol{k}_{j^*,\cdot,h^*})_{\epsilon_{2N},N}) \geq 0.95$$

Hence the optimal solution does represent a properly fitted function. Note that for any fixed $h$, the algorithm will find an optimal function in $\mathcal{L}^0_h(\Theta, \mathcal{K}) \bigcap \mathcal{L}^{(\epsilon^*)}_h$. Hence an algorithm to find an optimal function in the $h_{\max} > 1$, $h^*$ unknown, $h^* \in \mathcal{H}$ case could be designed as follows:

1. Divide $\mathcal{L}^0(\Theta, \mathcal{K})$ into its component classes $\mathcal{L}^0_1(\Theta, \mathcal{K}), \mathcal{L}^0_2(\Theta, \mathcal{K}), \ldots, \mathcal{L}^0_{h_{\max}}(\Theta, \mathcal{K})$.

2. On each class $\mathcal{L}^0_h(\Theta, \mathcal{K})$, $h = 1, \ldots, h_{\max}$, use an elitist GA to find $L(\boldsymbol{\theta}_{j^*,\cdot,h}, \boldsymbol{k}_{j^*,\cdot,h})$ where $FF_{\epsilon^*,N,\Theta,\mathcal{K}}(L(\boldsymbol{\theta}_{j^*,\cdot,h}, \boldsymbol{k}_{j^*,\cdot,h})) = \max_{L(\theta_{j,\cdot,h}, k_{j,\cdot,h}) \in \mathcal{L}^0_h(\Theta,\mathcal{K})} FF_{\epsilon^*,N,\Theta,\mathcal{K}}(L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}))$.

3. Define $L_{opt} = \{L(\boldsymbol{\theta}_{j^*,\cdot,1}, \boldsymbol{k}_{j^*,\cdot,1}), \ldots, L(\boldsymbol{\theta}_{j^*,\cdot,h_{\max}}, \boldsymbol{k}_{j^*,\cdot,h_{\max}})\}$. Then the solution is taken as the function $L(\boldsymbol{\theta}_{j^*,\cdot,h'}, \boldsymbol{k}_{j^*,\cdot,h'}) \in L_{opt}$ where

$$FF_{\epsilon^*,N,\Theta,\mathcal{K}}(L(\boldsymbol{\theta}_{j^*,\cdot,h'}, \boldsymbol{k}_{j^*,\cdot,h'})) = \max_{L(\theta_{j^*,\cdot,h}, k_{j^*,\cdot,h}) \in L_{opt}} FF_{\epsilon^*,N,\Theta,\mathcal{K}}(L(\boldsymbol{\theta}_{j^*,\cdot,h}, \boldsymbol{k}_{j^*,\cdot,h}))$$

This type of GA we call a *partitioned genetic algorithm.* We have noted previously that, theoretically, the elitist GA will converge to an optimal solution for a sufficiently large number of iterations. It is easy to show (although the proof will not be given here) that the proof of convergence to an optimal string holds for this algorithm as well. This justifies Statement #3 (see Section 4.3); having previously justified Statements #1 and #2, we conclude that our research goal can be achieved. Due to efficiency considerations, we plan to implement a partitioned GA instead of a VLGA. However, the optimal string of the partitioned GA will represent an optimal function - by the same reasoning as above and by its relationship to the variable length genetic algorithm.

Using a partitioned GA, for each fixed value of $h \in \mathcal{H}$ we find the string which maximizes

$$\sum_{i=1}^{N} \mathcal{I}_{\epsilon^*,L(\theta_{j,\cdot,h}, k_{j,\cdot,h})}(\mathrm{x}_i, \mathrm{y}_i)$$

i.e., the minimal number of points have distance greater than $\epsilon^*$ from the function represented by the chosen string. From the resulting group of strings we then select as our optimal string the one which represents the least number of lines. We observe that this optimal string will represent the function $L(\boldsymbol{\theta}_{j^*,\cdot,h'}, \boldsymbol{k}_{j^*,\cdot,h'}) \in \mathcal{L}^0(\Theta, \mathcal{K})$ which maximizes

$$FF_{\epsilon^*,N,\Theta,\mathcal{K}}(L(\boldsymbol{\theta}_{j,\cdot,h'}, \boldsymbol{k}_{j,\cdot,h'})) = \sum_{i=1}^{N} \mathcal{I}_{\epsilon^*,L(\theta_{j,\cdot,h'}, k_{j,\cdot,h'})}(\mathrm{x}_i, \mathrm{y}_i) + (1 - \frac{h'}{h_{\max}})$$

Noting the similarity between the fitness function of the partitioned GA and the fitness function of the VLGA (see Section 3.2, Eqn. 7), we conclude that an optimal string

from the partitioned GA will represent a set of lines that minimizes that number of points whose distance from the set of lines is greater than $\epsilon^*$ yet uses the least number of lines required to do so. Hence an optimal solution of the partitioned GA does represent an optimal function.

### 4.5.3 Remarks

1. We have justified the claim that for $h_{\max} > 1$, an optimal function is contained in the search space of our algorithm and that our algorithm will converge to an optimal solution, where an optimal solution represents an optimal function. We have also established that for $\Theta$ and $\mathcal{K}$ large and *critical level* $= 1 - crit = 1.0$, the solution of our algorithm will converge to the least squares solution (i.e., the optimal solution with $h^*$ lines) as the number of iterations and $N \to \infty$. Hence our algorithm can be used to fit both robust and non-robust solutions.

2. For $1 - crit < 1.0$, the optimal solution to which the algorithm converges may have $h'$ lines where $h' \neq h^*$. However, a value of $1 - crit < 1.0$ is useful when the given dataset contains outliers.

3. As in the $h_{\max} = 1$ case, for $h_{\max} > 1$ we do not know an appropriate value for $\epsilon$ apriori. Hence the proposed algorithm must include an adaptive procedure which will search for an appropriate value for $\epsilon$. In the next section the partitioned GA described above will be expanded to include this adaptive search.

4. A given dataset of size $N$ yields a corresponding estimate $B_N$ of the support $B$ of the probability density function of (X, Y). The proposed method, by estimating $f(x)$ via maximization of $FF_{\epsilon^*,N,\Theta,\mathcal{K}}$ where $FF_{\epsilon^*,N,\Theta,\mathcal{K}}$ depends upon the given dataset, essentially uses $B_N$ to estimate $f(x)$. Assuming that $\Theta$, $\mathcal{K}$, and the maximum number of iterations are chosen sufficiently large, given a sequence of datsets of sizes $N$, $N \to \infty$, we may characterize our estimation of $f(x)$ as a maximization over the corresponding sequence of subspaces $B_N$ which belong to the interior of $B$ and approach $B$ as $N$ increases. In this sense our maximization process is reminiscent of Grenander's *method of sieves* [34].

The theoretical foundation of our algorithm has been established. In the next section we show the results of implementing our algorithm on several datasets and discuss how these results compare to those of similar methods.

# 5 Experimental Results

## 5.1 Methods and Implementation

### 5.1.1 Genetic Algorithm

We applied to each dataset a partitioned GA with either $\mathcal{H} = \{2, 3, 4\}$ or $\mathcal{H} = \{3, 4, 5\}$ (we did not start with $h = 1$ because this choice for $h$ was clearly inappropriate).

We utilized binary coding although an alternate coding scheme could have been used. Since we chose values for $\Theta$ and $\mathcal{K}$ which were fixed but very large (by specifying $l_a = 8$ and $l_d = 12$), following remark 2 of Section 4.4.3 we implemented a partitioned GA which adaptively searched only for $\epsilon$. We refer to such a GA as a *variable epsilon* genetic algorithm. For each value $h \in \mathcal{H}$, the variable epsilon genetic algorithm can be described as follows:

1. Set the global parameters for population size $M$ ($M = 50$), crossover probability $p_c$ ($p_c = 0.8$), number of characters for representing angle $l_a$ ($l_a = 8$), number of characters for representing distance or offset value $l_d$ ($l_d = 12$), and the maximum number of iterations $MaxNit$ ($MaxNit = 20000$). The selection for $M$ depends on the computing power of the machine while $l_a$ and $l_d$ depend upon the desired precision of our result. The mutation probability $p_m$ was varied as a function of the number of iterations - see Section 3.2 and [29] for more details. Further comments on parameter value selection can be found in Section 6.

2. Choose the *critical level* = 1 - *crit* = percentage of data points to fall within $\epsilon$ of the final fitted piecewise linear function, and a large initial value for $\epsilon$. The fitness value of each string (where each string represents a function $L(\boldsymbol{\theta}_{j,\cdot,h}, \boldsymbol{k}_{j,\cdot,h}) \in \mathcal{L}^0(\Theta, \mathcal{K})$) is determined by Equation 10.

3. Run an elitist GA until either (1) $(1 - crit) \leq$ (the maximum fitness value of the population)/(number of observations), or (2) the maximum number of iterations, $MaxNit$, is reached. If (1) occurs, then set $\epsilon = \epsilon - 0.01$, $Nit$ = iteration number $= 1$, and restart the elitist GA. If (2) occurs, report the function corresponding to the string with the maximum fitness value as the final result for the given value of $h$.

4. Once the algorithm has been executed for each possible value of $h$, compare the results across $h$ values and select the string corresponding to the overall maximum fitness value as the optimum string.

Note that within each run of our algorithm the value of $h$ is **fixed**. We then compare the results of each run of the algorithm for each value of $h \in \mathcal{H}$ to determine the overall optimal string.

### 5.1.2 Data

The proposed algorithm was applied to the four generated datasets described in Table 1. Each dataset has normally distributed noise (denoted $Nor$(mean, sd)) and was designed with specific characteristics - dataset #1: no outliers, equally spaced knots; dataset #2: no outliers, unequally spaced knots; dataset #3: outliers, equally spaced knots; and dataset #4: outliers, unequally spaced knots. The datasets range in size from 50 to 375 data points.

| Set # | $N$ | $h^*$ | Generating Function | Noise |
|---|---|---|---|---|
| 1 | 375 | 3 | $g(x) = \begin{cases} -2.0x & -1.0 \leq x < 0.0 \\ 0.5x & 0.0 \leq x < 1.0 \\ -0.5x + 1.0 & 1.0 \leq x \leq 2.0 \end{cases}$ | $Nor(0, 0.25)$ |
| 2 | 320 | 3 | $g(x) = \begin{cases} 3.88x + 10.44 & -3.0 \leq x < -1.29 \\ -1.74x + 3.14 & -1.29 \leq x < 3.7 \\ 3.77x - 17.25 & 3.7 \leq x \leq 4.9 \end{cases}$ | $Nor(0, 1)$ |
| 3 | 60 | 3 | $g(x) = \begin{cases} -0.1x + 0.6 & 0.0 \leq x < 1.0 \\ -1.1x + 1.6 & 1.0 \leq x < 2.0 \\ 0.8x - 2.2 & 2.0 \leq x \leq 3.0 \end{cases}$ | $Nor(0, 0.1)$ |
| 4 | 50 | 4 | $g(x) = \begin{cases} 1.3x + 2 & 0.0 \leq x < 0.9 \\ 0.05x + 3.125 & 0.9 \leq x < 2.1 \\ -0.4x + 3.2 & 2.1 \leq x < 4.3 \\ 3.2x - 11.41 & 4.3 \leq x \leq 5.0 \end{cases}$ | $Nor(0, 0.2)$ |

Table 1: Experimental Datasets

### 5.1.3 Comparison

For each dataset the results of the variable epsilon genetic algorithm were compared to the results of three other piecewise linear fitting methods:

1. *b-spline*: a b-spline of degree 2 fit using the functions **bs** and **lm** in the software package S-plus, version 3.3, release 1 [33]. The knot locations are equally spaced by default.

2. *Boor spline*: The library **pppack** containing functions from de Boor [13] was downloaded from the Netlib repository (http://www.netlib.org/). The function **l2main** was implemented to fit a b-spline of degree 2 whose knot locations were improved by the subroutine **newnot**. The number of iterations was set at 900.

3. *least-squares GA*: a genetic algorithm which has the same global parameters as the variable epsilon GA but which fits a piecewise linear function by minimizing the sum of squared error.

All three methods were applied to each dataset for each value $h \in \mathcal{H}$. Then for each method, the results for all $h$ values were compared and the best result was chosen as the method's overall solution. We have not attempted to compare results under neural network based methods since we are fitting straight lines and not curves. We did not compare our results with those of other GA based methods [2, 8, 26] because the limited foci of these works were incompatible with that considered here.

All experiments were run on a Sun Sparcstation 5.

Due to space constraints, for each dataset we discuss only selected results which demonstrate the relative strengths and weaknesses of our method. In the figures and tables, the phrase 'GA ls' refers to the least squares GA while the phrase 'GA eps' refers to the variable epsilon GA.

## 5.2 Selected Results

### 5.2.1 Dataset 1

This was our 'nice' dataset - no outliers and a generating function with equally spaced knots. Since there were no outliers, we used the sum of squared error (SSE) as a standard of comparison. The choice of *critical level* for the variable epsilon GA was 95%. All four methods chose the correct number of lines; the least squares GA yielded the best fit, followed by the b-spline, the variable epsilon GA, and the Boor spline, respectively. We observe in Figure 5 that only the Boor spline appears to be unsatisfactory. The run times for the splines were almost instantaneous while the GAs took longer (approx. 20 minutes). It should be noted, however, that with $h$=3 our string length was 60, hence the number of possible strings was $2^{60} \approx 1152921 * 10^{12}$! Despite this, the variable epsilon GA yielded an acceptable result within the initial 5 minutes of CPU time.

| Method | SSE | Method | SSE |
|---|---|---|---|
| b-spline | 22.26 | Boor spline | 34.23 |
| GA ls | 17.89 | GA eps | 23.73 |

Table 2: SSE values for Dataset 1, h=3

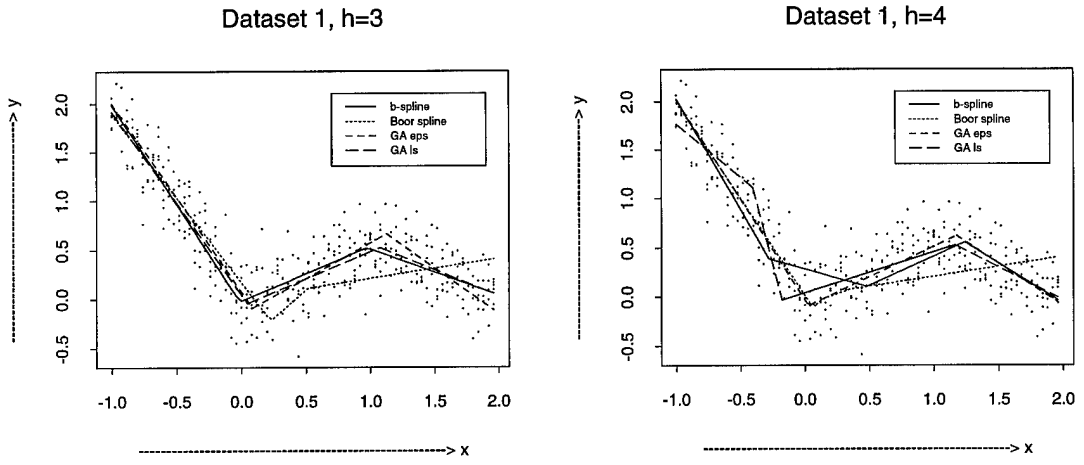



Figure 5: Results of Dataset 1 for $h$=3 and $h$=4

We observed that although the variable epsilon GA did not yield the best fit, it was the only method which yielded a very nice fit (SSE = 22.80) when the number of pieces was misspecified - when $h$ was set equal to 4 although the generating function has 3 pieces ($h^*$=3)(see Figures 5 and 6). The algorithm was almost able to merge 2 of the pieces into 1. Note that in Figure 6, where the result of applying the variable epsilon GA on dataset 1 is shown for $h = 4$, it is very difficult to see 4 lines with the naked eye. Hence the variable epsilon GA was more robust against misspecification of $h$ in relation to the other methods.

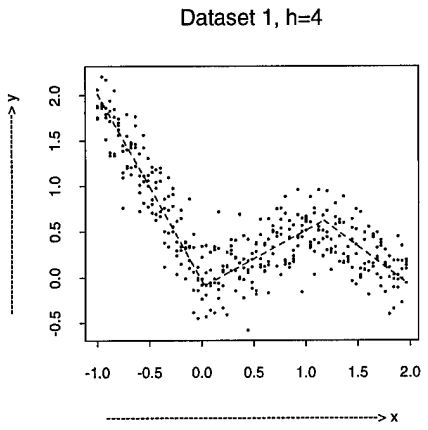



Figure 6: Result of eps GA on Dataset 1 with $h$=4

### 5.2.2 Dataset 2

While dataset 1 had a generating function with equally spaced knots, this dataset had a generating function with unequally spaced knots - the middle piece was considerably longer than either of the end pieces. For this reason the fit of the b-spline was quite poor in comparison to the genetic algorithms' results. However, why the fit of the Boor spline, which varies the knot locations, was so poor remains a mystery. The best model from each method had the correct number of lines. The variable epsilon GA (with $1 - crit$=95%) yielded the best model, followed by the least squares GA model, the b-spline, and the Boor spline.

The failure of the least squares GA to provide the lowest SSE model was most likely due to insufficient iterations. Again, the speed of the spline algorithms was almost instantaneous, while the GA algorithms took about 5 minutes of CPU time to yield an acceptable result and about 15 minutes of CPU time to complete the maximum number of iterations.

With this dataset, only the Boor spline failed to merge lines for a better fit when the choice of $h$ was large. This suggests that the least squares GA would have also merged lines for dataset 1 but perhaps more iterations of the algorithm were required. The b-spline could not match the quality of fit of the GAs because its knot locations were fixed.

| Method | SSE | Method | SSE |
|---|---|---|---|
| b-spline | 610.24 | Boor spline | 836.31 |
| GA ls | 415.35 | GA eps | 351.57 |

Table 3: SSE values for Dataset 2, h=3

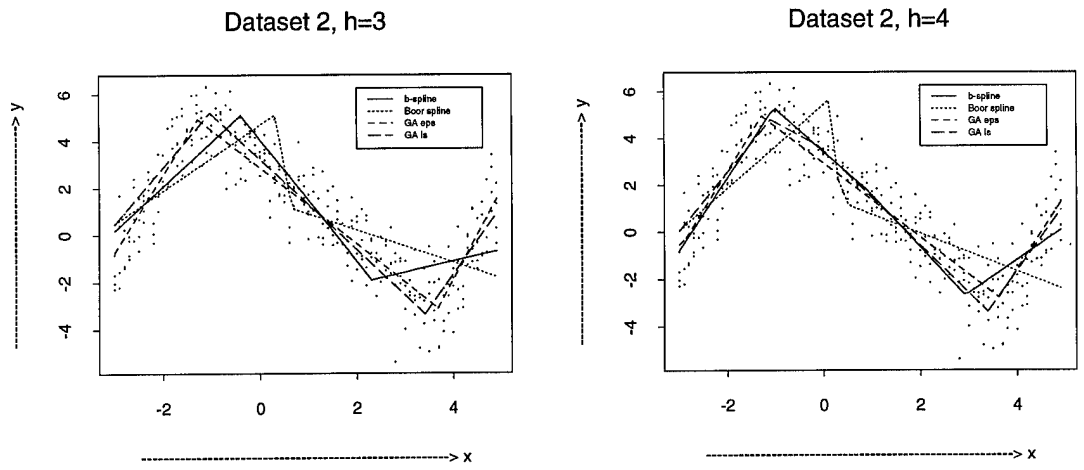



Figure 7: Results of Dataset 2 for $h = 3$ and $h = 4$

### 5.2.3 Dataset 3

The other strengths of the variable epsilon GA became evident when we considered datasets with outliers. Dataset 3 had equally spaced knots and contained a cluster of 6 outliers. The outliers made the SSE criterion useless as a measure of fit, so we based our statements about the quality of the results on visual comparisons. Figure 8 clearly shows that the outliers caused difficulties for all methods except for the variable epsilon GA. The b-spline and the least squares GA showed particularly poor results. However, the parameter *crit* of the variable epsilon GA made it robust against outliers. $1 - crit$ was set at 90% to allow for the outliers - leading, in this case, to a superior fit.

If $1 - crit$ had been set at 95%, as was done previously, the variable epsilon GA result would have been closer to that of the least squares GA (results omitted due to limited space). Even if the outliers had not been clustered, the variable epsilon GA would have reached a superior solution - see dataset 4 for an example with non-clustered outliers.

As with dataset 1, only the variable epsilon GA yielded a very nice fit when the number of lines was misspecified as $h = 4$ instead of $h = 3$ (see Figures 8 and 9). Despite solution spaces with sizes of order $2^{60} \approx 1152921 * 10^{12}$ for $h$=3, $2^{80} \approx 1208925 * 10^{18}$ for $h = 4$, and $2^{100} \approx 1267650 * 10^{24}$ for $h$=5, the variable epsilon GA was able to yield a near-optimal result in about 20 minutes. This can be seen in Figure 9 where the result of the variable epsilon GA for $h = 3$ and the generating lines of Dataset 3 have been plotted for comparison.

### 5.2.4 Dataset 4

Our final dataset combined unequally spaced knots with the presence of 4 outliers which were not clustered. We set $1 - crit$=92%. From Figure 10 it is clear that only the variable epsilon GA provided a reasonable fit. The Boor spline and the least

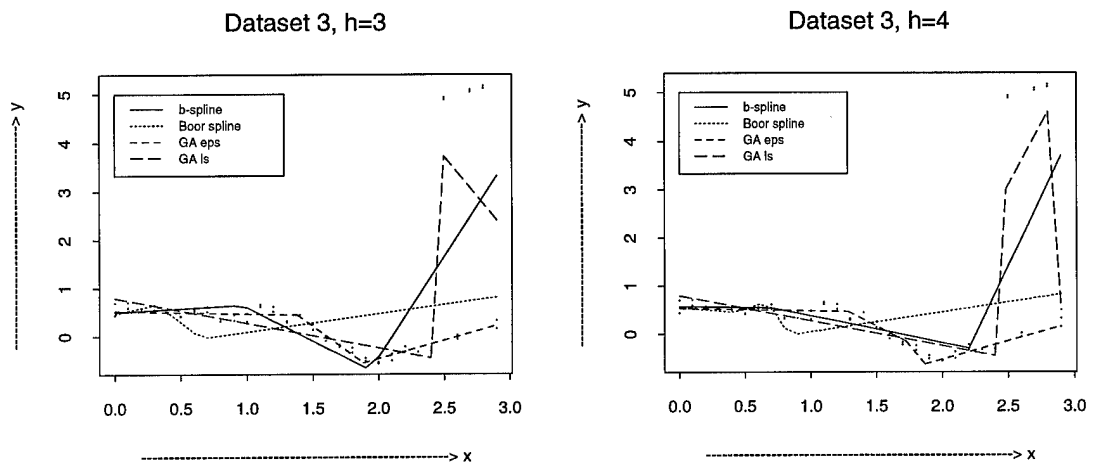



Figure 8: Results of Dataset 3 for $h = 3$ and $h = 4$

squares GA were adversely affected by the outliers, while the b-spline failed to capture the shape of the dataset. Again, the robustness of the variable epsilon GA proved essential for a proper fit. We also found that when the specified number of lines was large ($h = 4$ instead of $h = 3$) the variable epsilon GA was able to compensate for this by selecting pieces which almost merged.

## 5.3 Conclusions

The above results demonstrate that for 'nice' datasets (no outliers) the variable epsilon GA can provide a fit comparable to those of a least squares GA or splines. Although the variable epsilon GA did not always yield the 'best' result for nice datasets, it always achieved a satisfactory fit very quickly. When the number of pieces was large, the algorithm did a excellent job of decreasing the effective number of pieces by choosing lines which almost merged. By increasing the number of iterations and/or the string length, it is likely that even better results can be achieved.

For datasets with outliers, the variable epsilon GA is capable of yielding results far superior to those of comparable methods. As with 'nice' datasets, if too many pieces have been specified the algorithm can almost merge pieces to yield a model with the appropriate number of effective knots. Despite large search spaces (e.g., on the order of $2^{60}$ and $2^{80}$) the algorithm can reach an acceptable solution in about 5 minutes and a near-optimal solution in about 20 minutes. The variable *crit* makes it possible to adjust for outliers while the variables $\Theta$ and $\mathcal{K}$ provide some control over the precision of the final fit. By allowing $\epsilon$ to be determined adaptively, our algorithm can find a result which (1) satisfies the critical value and (2) is closest to the majority of the data points with respect to other solutions.

Thus the effectiveness of the variable epsilon GA has been conclusively demonstrated for fitting both robust and non-robust piecewise linear functions.

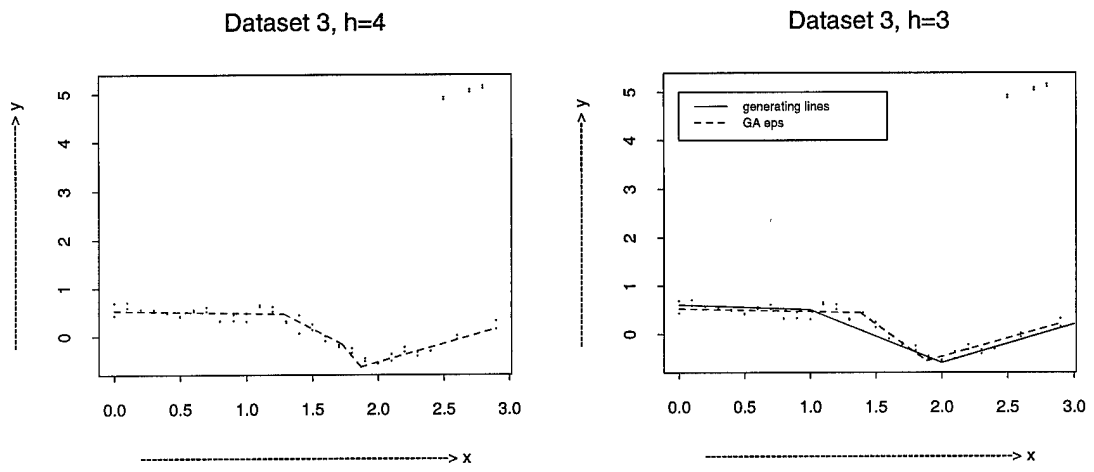



Figure 9: Result of eps GA on Dataset 3 with $h$=4; Comparison of eps GA with generating lines on Dataset 3 with $h$=3

## 5.4 Remarks

1. Since the solution given by the variable epsilon GA is the result of a random process, we decided to run an experiment several times on dataset 5 and examine the variability of the results. Dataset 5 contained several outliers, had equally spaced knots, and met the specifications in Table 4.

| Set # | $N$ | $h$ | Function | Noise |
|---|---|---|---|---|
| 5 | 60 | 3 | $g(x) = \begin{cases} 2.5x & 0.0 \leq x < 1.0 \\ -4.0x + 6.5 & 1.0 \leq x < 2.0 \\ 3.5x + -8.5 & 2.0 \leq x \leq 3.0 \end{cases}$ | $Nor(0, 0.1)$ |

Table 4: Dataset 5

A variable epsilon GA was executed 8 times with the global parameters set as in Section 5.1.1 and $1 - crit = 95\%$. The results are shown in Figure 11 and Table 5.

Figure 11 does not appear to contain 8 functions because several results were identical. By examining the ranges of both the slopes and intercepts of the linear pieces, the achieved level of consistency appears to be satisfactory. Table 5 shows the ranges of parameter values of all fitted lines. For example, the slope and intercept of the first generating line are 2.5 and 0, respectively, and the experiments generated corresponding ranges of (2.41421, 3.02704) and (-0.188247, 0.180099). Further research is needed to determine a more precise measure of variability of results.

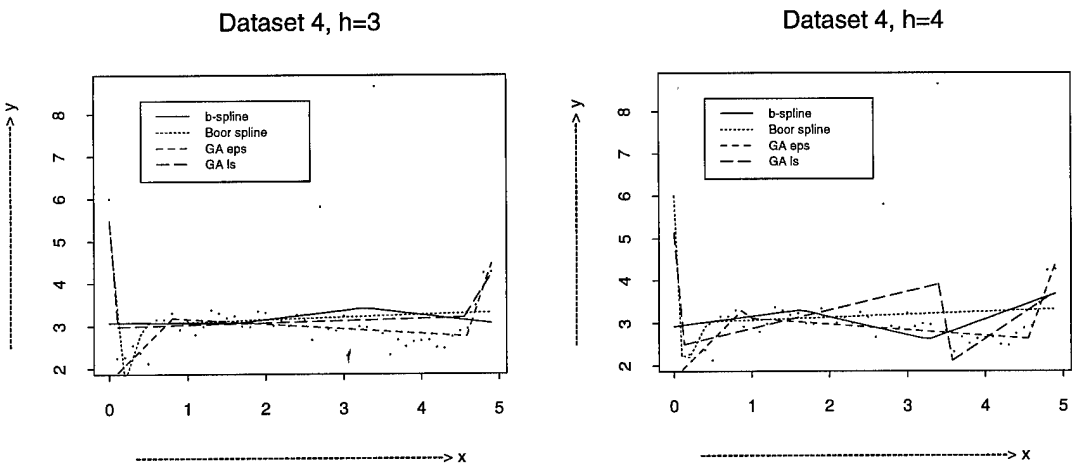



Figure 10: Results of Dataset 4 for $h = 3$ and $h = 4$

| | Piece #1 | Piece #2 | Piece #3 |
|---|---|---|---|
| Intercept | (-0.188247, 0.180099) | (6.64789, 6.91933) | (-8.73460, -7.92068) |
| Slope | (2.41421, 3.02704) | (-4.21080, -3.99222) | (3.29656, 3.61354) |

Table 5: Dataset 5 Results; Ranges for Intercepts and Slopes

2. For our experiments we considered datasets where $h^*$ was either 3 or 4 and $h$ was between 2 and 5. However, the values for $h^*$ and $h$ that can be considered can be increased simply by increasing the available computational resources. Note that when $h$=4, the string length is 80 so the size of the solution space is on the order of $2^{80}$. Yet we were able to achieve near-optimal results in 20 minutes. With greater resources, a larger solution space could most likely be handled in approximately the same CPU time.

# 6 Conclusions and Future Research

By using genetic algorithms we have devised a method for fitting piecewise linear functions to datasets in $\mathcal{R}^2$ which not only optimizes the number of pieces but also optimizes the knot locations. With the assumption that the probability density function of our random variables is symmetric, the above theory shows that our method will lead to a piecewise linear function which 'fits' the given dataset. However, even if we do not make this assumption (so our only assumption about the data is that the underlying probability density function is continuous), our method will still yield an optimal fit for the given dataset, optimal with respect to the chosen fitness function.

The proposed method yielded very good results in the presence of noise. The parameter *crit* makes it possible for our algorithm to reach a near-optimal result even in the presence of outliers. The convergence of genetic algorithms has been shown for practically any choice of parameter values (e.g., $M = 50$, $p_c = 0.8$, etc.) although the

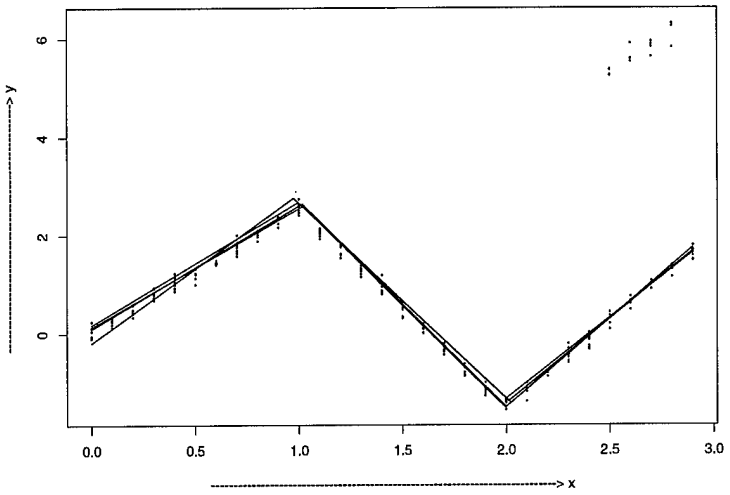



Figure 11: Example of Variability of Results

best choices are still a matter of contention. The formulation of an optimal stopping rule is a subject of ongoing research, although it is known that increasing the number of iterations leads to a result with better accuracy.

As mentioned previously, we would like to develop a method for providing confidence limits for our results. It may also be possible to decrease CPU time by developing a heuristic for determining a 'best' initial value for $\epsilon$. The closer $\epsilon$ can be chosen to its theoretical optimum, the less iterations and hence less CPU time will be required by the algorithm. Given the speed of the proposed method we believe it is feasible to extend the use of genetic algorithms to fitting curvilinear models and to datasets of dimension greater than 2.

# 7 Acknowledgements

The research of Ms. J. Pittman is supported by the Army Research Office under Assert grant # DAAG55-97-1-0219.

The research of C.A. Murthy is supported by the Army Research Office under Assert grant # DAAHO4-96-1-0082.

# 8 Appendix

**Proof of Theorem 4.1:**

Let $\epsilon \geq \epsilon^*$, $\Theta$, and $\mathcal{K}$ be given.
Suppose $\exists\ L(\theta_{j^*,1,1}, k_{j^*,1,1}) :\ L(\theta_{j^*,1,1}, k_{j^*,1,1}) \in\ \mathcal{L}^{(\epsilon)}$ and

$$\{(x,y) : y = L(\theta_{j^*,1,1}, k_{j^*,1,1})(x),\ x \in [z_{11}, z_{21}]\} \bigcap\ rect = \emptyset$$

Then either

$$\{(x,y) : y \in\ [L(\theta_{j^*,1,1}, k_{j^*,1,1})(x), L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) + \epsilon],\ x \in [z_{11}, z_{21}]\} \bigcap\ rect = \emptyset$$

or

$$\{(x,y) : y \in\ [L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) - \epsilon, L(\theta_{j^*,1,1}, k_{j^*,1,1})(x)],\ x \in [z_{11}, z_{21}]\} \bigcap\ rect = \emptyset$$

Assume without loss of generality that

$$\{(x,y) : y \in\ [L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) - \epsilon, L(\theta_{j^*,1,1}, k_{j^*,1,1})(x)],\ x \in [z_{11}, z_{21}]\} \bigcap\ rect = \emptyset$$

This implies that $\frac{1}{N}\sum_{i=1}^{N} \varphi((\mathrm{x}_i, \mathrm{y}_i)) \geq 0.95$, where

$$\varphi((x,y)) = \begin{cases} 1 & (x,y) \in \{(x,y) :\ y \in [L(\theta_{j^*,1,1}, k_{j^*,1,1})(x), L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) + \epsilon],\ x \in [z_{11}, z_{21}]\} \\ 0 & \text{otherwise} \end{cases}$$

Let $(\bar{\bar{\mathrm{x}}}, \bar{\bar{\mathrm{y}}}) = (\sum_{i=1}^{N} \varphi((\mathrm{x}_i, \mathrm{y}_i)))^{-1} \sum_j (\mathrm{x}_j, \mathrm{y}_j)$ where the sum is taken over all
$(\mathrm{x}_j, \mathrm{y}_j) \in \{(x,y) :\ y \in [L(\theta_{j^*,1,1}, k_{j^*,1,1})(x), L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) + \epsilon], x \in [z_{11}, z_{21}]\}$.

Define $L(\theta_{j^{**},1,1}, k_{j^{**},1,1}) : L(\theta_{j^{**},1,1}, k_{j^{**},1,1})$ is parallel to $L(\theta_{j^*,1,1}, k_{j^*,1,1})$ and $\bar{\bar{\mathrm{y}}} = L(\theta_{j^{**},1,1}, k_{j^{**},1,1})(\bar{\bar{\mathrm{x}}})$.

Then
$\{(x,y) : y \in [L(\theta_{j^*,1,1}, k_{j^*,1,1})(x), L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) + \epsilon],\ x \in [z_{11}, z_{21}]\} \subseteq E_{\epsilon, L(\theta_{j^{**},1,1}, k_{j^{**},1,1})}$

Thus $L(\theta_{j^{**},1,1}, k_{j^{**},1,1}) \in \mathcal{L}^{(\epsilon)}$ and since $\bar{\bar{\mathrm{y}}} = L(\theta_{j^{**},1,1}, k_{j^{**},1,1})(\bar{\bar{\mathrm{x}}})$,

$$\{(x,y) : y = L(\theta_{j^{**},1,1}, k_{j^{**},1,1})(x),\ x \in [z_{11}, z_{21}]\} \bigcap rect \neq \emptyset$$

♠

For a graphical representation see Figure 12a. Note that 95% of the datapoints fall within $\epsilon$ of $L(\theta_{j^*,1,1}, k_{j^*,1,1})$ while 95% of the datapoints fall within $\epsilon/2$ of $L(\theta_{j^{**},1,1}, k_{j^{**},1,1})$.

**Proof of Proposition 4.1:**

Let $L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{P}_1$ and $\xi > 0$ be given. Choose $\Theta_\xi : \pi/2^{l_a} = \pi/\Theta_\xi < \xi/2$. Similarly, choose $\mathcal{K}_\xi : \delta = diag/(2^{l_d} - 1) = diag/(\mathcal{K}_\xi - 1) < \xi/2$. Then $\exists\ L(\theta, k) \in \mathcal{L}_1^0(\Theta_\xi, \mathcal{K}_\xi) : 2.$ is satisfied. Since for any $h$, $h = 1, \ldots, h_{\max}$, $\{\mathcal{L}_h^0(\Theta, \mathcal{K}) : l_a = 1, 2, \ldots\}$ and $\{\mathcal{L}_h^0(\Theta, \mathcal{K}) : l_d = 1, 2, \ldots\}$ represent increasing sequences of nested sets, if $L(\theta, k) \in \mathcal{L}_1^0(\Theta_\xi, \mathcal{K}_\xi)$, then $L(\theta, k) \in \mathcal{L}_1^0(\Theta, \mathcal{K})\ \forall\ \Theta > \Theta_\xi$ and $\forall\ \mathcal{K} > \mathcal{K}_\xi$. Hence *1* is satisfied. ♠

**Proof of Theorem 4.2:**

Let $\xi > 0$ be given. Choose $\Theta_\xi : \pi/2^{l_a} = \pi/\Theta_\xi < \xi/2$. Then for $\theta_{\xi_{(i)}} \in \{\frac{\pi}{2^{\Theta_\xi}}, \ldots, \pi\};\ \theta_{\xi_{(i)}} \leq \theta_{\xi_{(i+1)}}\ \forall\ i,\ i = 1, \ldots, \Theta_\xi - 1\}$, we have $|0 - \theta_{\xi_{(1)}}| < \xi/2$,
$|\theta_{\xi_{(1)}} - \theta_{\xi_{(2)}}| < \xi/2, \ldots, |\theta_{\xi_{(\Theta_\xi)}} - \pi| < \xi/2$. So given any $L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{P}_1$ we can choose $\Theta_\xi$ so that $\exists\ \theta_{\xi_{(i)}} \in \{\frac{\pi}{2^{\Theta_\xi}}, \ldots, \pi\} : |\theta_{m,1,1} - \theta_{\xi_{(i)}}| < \xi/2$.

For any angle $\theta_{\xi_{(n)}} \in [\frac{n\pi}{\Theta_\xi}, \frac{(n+1)\pi}{\Theta_\xi}],\ n = 1, \ldots, \Theta_\xi - 1$ , the corresponding $\rho_{\xi_{(n)}} \in [\gamma_{\xi_{(n_1)}}, \gamma_{\xi_{(n_2)}}],\ l_{\theta_{\xi_{(n)}}} \leq \gamma_{\xi_{(n_1)}} \leq \gamma_{\xi_{(n_2)}} \leq diag$. Find

$$\nu = \max_n \{\sup_{\xi_{(n)}} \{[\gamma_{\xi_{(n_2)}} - \gamma_{\xi_{(n_1)}}],\ n = 1, \ldots, \Theta_\xi - 1\}\}$$

Choose $\mathcal{K}_\xi : \nu/\mathcal{K}_\xi < \xi/4$ and $\mathcal{K}_\xi = 2^y$ for some $y \in \mathcal{R}$. Then given any $L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{P}_1$ we can choose $\mathcal{K}_\xi$ so that $\exists\ k_{\xi_{(i)}} \in \{0, \ldots, 2^{\mathcal{K}_\xi} - 1\} : |\rho_{\xi_{(i)}} - \rho_{m,1,1}| < \xi/2$.

Hence given any $\xi > 0$ and $L(\theta_{m,1,1}, k_{m,1,1}) \in \mathcal{P}_1$ we can find $\Theta_\xi$ and $\mathcal{K}_\xi$ so that $\exists\ L(\theta_{\xi_{(i)}}, k_{\xi_{(i)}}) \in \mathcal{L}_1^0(\Theta_\xi, \mathcal{K}_\xi) : |\theta - \theta_{m,1,1}| < \xi/2$ and $|\rho_{m,1,1} - \rho| < \xi/2$.

If $L(\theta_{\xi_{(i)}}, k_{\xi_{(i)}}) \in \mathcal{L}_1^0(\Theta_\xi, \mathcal{K}_\xi)$, then $L(\theta_{\xi_{(i)}}, k_{\xi_{(i)}}) \in \mathcal{L}_1^0(\Theta, \mathcal{K})\ \forall\ \Theta > \Theta_\xi$ and $\forall\ \mathcal{K} > \mathcal{K}_\xi$. Hence *1.* and *2.* are satisfied.♠

**Proof of Proposition 4.2:**

Define $\mathcal{S}_1 = \bigcup_{i=1}^{\infty} \mathcal{L}_1^0(\Theta_i, \mathcal{K}_i)$. Note $\mathcal{P}_1 \subseteq \mathcal{S}_1$. Note that $\mathcal{S}_1 \subseteq \mathcal{T}_1$ so $\mathcal{P}_1 \subseteq \mathcal{T}_1$. Since we are requiring our proper lines which pass through *rect*, the proper lines in $\mathcal{P}_1 \equiv$ the proper lines in $\mathcal{T}_1$.♠

**Proof of Theorem 4.3**

Note that $\mathcal{L}_1^0(\Theta_i, \mathcal{K}_i) \subset \mathcal{L}_1^0(\Theta_{i'}, \mathcal{K}_{i'})\ \forall\ i' > i$. Hence $\mathcal{A}_{\epsilon i} \subseteq \mathcal{A}_{\epsilon i'}\ \forall\ i' > i$. Thus $\mathcal{A}_{\epsilon \lim} = \lim_{i \to \infty} \mathcal{A}_{\epsilon i}$ exists [30]. But $\mathcal{A}_{\epsilon \lim}$ is the set of optimal lines in $\mathcal{S}_1$ where $\mathcal{P}_1 \subseteq \mathcal{S}_1 \subseteq \mathcal{T}_1$. By Proposition 4.2, $\mathcal{A}_{\epsilon \lim} = \mathcal{A}_\epsilon$. ♠

**Proof of Theorem 4.4**

Suppose $\exists\ L(\theta_{j^*,1,1}, k_{j^*,1,1}),\ L(\theta_{j^{**},1,1}, k_{j^{**},1,1}) \in \mathcal{A}_{\epsilon_{1.0}^*}(1.0)$ :
$L(\theta_{j^*,1,1}, k_{j^*,1,1}) \neq L(\theta_{j^{**},1,1}, k_{j^{**},1,1})$. Then for all possible values of (x, y)

$$(x, y) \in \{(x, y) : y \in [L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) - \epsilon_{1.0}^*,\ L(\theta_{j^*,1,1}, k_{j^*,1,1})(x) + \epsilon_{1.0}^*]\}$$

and

$$(\mathrm{x}, \mathrm{y}) \in \{(x,y) : y \in [L(\theta_{j^{**},1,1}, k_{j^{**},1.1})(x) - \epsilon^*_{1.0},\ L(\theta_{j^{**},1,1}, k_{j^{**},1,1})(x) + \epsilon^*_{1.0}]\}\ .$$

Recall that for $h^* = 1$, the support of $\alpha_1(x,y)$ was defined as $B$ where

$$B = \{(x,y) : y \in [\tfrac{d_1 - x\cos\theta_1}{\sin\theta_1} - \epsilon_0, \tfrac{d_1 - x\cos\theta_1}{\sin\theta_1} + \epsilon_0]\} \quad \text{for} \quad x \in [z_{11}, z_{21}]$$

As $N \to \infty$, $\epsilon^*_{1.0} \to \epsilon_0$ since $\epsilon^*_{1.0}$ is minimal and $FF_{\epsilon,N}$ is continuous for all $\epsilon$. Hence $\{(x,y) : y = L(\theta_{j^*,1,1}, k_{j^*,1,1})(x)\} \to \{(x,y) : x\cos\theta_1 + y\sin\theta_1 = d_1\}$ and $\{(x,y) : y = L(\theta_{j^{**},1,1}, k_{j^{**},1,1})(x)\} \to \{(x,y) : x\cos\theta_1 + y\sin\theta_1 = d_1\}$ Thus $L(\theta_{j^*,1,1}, k_{j^*,1,1}) = L(\theta_{j^{**},1,1}, k_{j^{**},1,1})$. ♠

See Figure 12b for a graphical representation of Theorem 4.4.

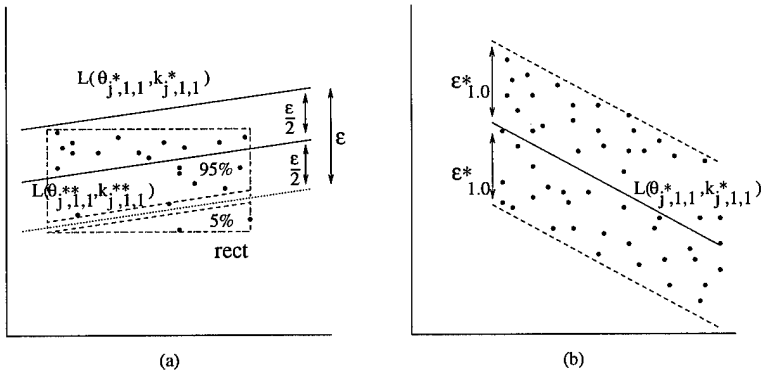



Figure 12: Graphical representations of (a) Theorem 4.2 and (b) Theorem 4.4

**Proof of Theorem 4.5:**

Given the continuity of $FF_{\epsilon,N}$ we know that as $N \to \infty$, for each $N$ large we may find $\epsilon_{1N}, \epsilon_{2N}, \epsilon_{3N}$, $0 < \epsilon_{1N} < \epsilon_{2N} < \epsilon_{3N}$, such that

- $AFF_{\epsilon_{1N},N}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon_{1N},N}) < 0.95$
- $AFF_{\epsilon_{2N},N}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon_{2N},N}) \geq 0.95$
- $AFF_{\epsilon_{3N},N}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon_{3N},N}) \geq 0.97$

Note that $L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon_{1N},N} \notin \mathcal{L}^{(\epsilon_{1N})}$ while $\epsilon_{3N}$ is not minimal, i.e., there exists some $\epsilon < \epsilon_{3N} : \mathcal{L}^{(\epsilon)} \neq \emptyset$. Hence for our stated goal (see Eqn. 11) we are interested only in $\epsilon_{2N}$. For each $N$ there may be infinitely many such $\epsilon_{2N}$. For a given $N$ and $\epsilon$ let one such $\epsilon_{2N}$ be $\epsilon^*_{2N}$. Then we conclude

$$\liminf_{N\to\infty} AFF_{\epsilon^*_{2N},N}(L(\theta_{j^*,1,1}, k_{j^*,1,1})_{\epsilon^*_{2N},N}) \geq 0.95$$

for appropriate $\Theta$ and $\mathcal{K}$. ♠

# References


[1] P.A. Kokkonis and V. Leute, "Least squares splines approximation applied to multicomponent diffusion data", *Computational Materials Science*, **6**, 1, pp.103-111, July 1996.

[2] C.L. Karr and B. Weck, "Improved Computer Modelling Using Least Median Squares Curve Fitting and Genetic Algorithms", *Fluid/Practice Separation Journal*, **8**, 2, pp.117-124, June 1995.

[3] R.L. Eubank, *Spline Smoothing and Nonparametric Regression*, 1st Edition. New York: Marcel Dekker, 1988.

[4] E. Mammen and S. Van de Geer, "Locally adaptive regression splines", *Annals of Statistics*, **25**, 1, pp.387-413, Feb. 1997.

[5] B. Cheng and D.M. Titterington, "Neural networks: A review from a statistical perspective", *Statistical Science*, **9**, pp.2-30, Feb. 1994.

[6] K. Warwick and R. Craddock, "An introduction to radial basis functions for system identification a comparison with other neural network methods", *Proc. of the IEEE Conf. on Decision and Ctrl*, **1**, pp.464-469, Dec. 1996.

[7] M. Werman and Z. Geyzel, "Fitting a Second Degree Curve in the Presence of Error.", *IEEE PAMI*, **17**, 2, p.207, Feb. 1995.

[8] C. L. Karr, D.A. Stanley, and B.J. Scheiner, "Genetic algorithm applied to least squares curve fitting", Report of investigations # 9339, U.S. Dept. of the Interior, Bureau of Mines, 1991.

[9] S. Chatterjee, M. Laudato, and L.A. Lynch, "Genetic algorithms and their statistical applications: an introduction", *Computational Statistics and Data Analysis*, **22**, 6, pp.633-651, Oct. 1996.

[10] L. Davis (ed.), *Handbook of Genetic Algorithms*, 1st Edition. New York: Van Nostrand Reinhold, 1991.

[11] C.A. Murthy and J. Pittman, "Optimal line fitting using genetic algorithms", *Pattern Recognition*, submitted, August 1997.

[12] D. Chen and R. Jain, "A robust back propagation learning algorithm for function approximation", *IEEE Trans. on Neural Networks*, **5**, pp.467-479, May 1994.

[13] C. De Boor, *A practical guide to splines*, 1st Edition. New York: Springer-Verlag, 1978.

[14] H.J. Larson, "Least squares estimation of linear splines with unknown knot locations", *Computational Statistics and Data Analysis*, **13**, pp.1-8, Jan. 1992.

[15] B.D. Ripley, *Pattern Recognition and Neural Networks*, 1st Edition. Cambridge University Press (1996).

[16] R.L. Eubank and P. Speckman, "Curve fitting by polynomial-trigonometric regression", *Biometrika*, **77**, 1, pp.1-9, March 1990.

[17] T.B. Nguyen and B.J. Oommen, "Moment-preserving piecewise linear approximations of signals and images.", *IEEE PAMI*, **19**, 1, pp.84-91, Jan. 1997.

[18] P.L. Rosin, "Techniques for assessing polygonal approximations of curves.", *IEEE PAMI*, **19**, 6, pp.659-666, June 1997.

[19] H. Schwetlick and T. Schütze, "Least squares approximation by splines with free knots", *BIT*, **35**, pp.361-384, Sept. 1995.

[20] P. Suchomski, "Method of optimal variable-knot spline interpolation in the l—2 discrete norm.", *International Journal of Systems Science*, **22**, 11, pp.2263-2274, Nov. 1991.

[21] W.S. Cleveland and S.J. Devlin, "Locally Weighted Regression: An approach to regression analysis by local fitting", *J. Amer. Statist. Assoc.*, **83**, 403, pp.596-610, Sept. 1988.

[22] L. Breiman, comment to B. Cheng and D.M. Titterington, "Neural networks: A review from a statistical perspective", *Statistical Science*, **9**, pp.38, Feb. 1994.

[23] D.G.T. Denison, B.K. Mallick, and A.F.M. Smith, "Automatic Bayesian Curve Fitting", Preprint, March 1996.

[24] B.K. Mallick, "Bayesian curve estimation by polynomials of random order", Preprint, April 1996.

[25] D. Bhandari, C.A. Murthy, and S.K. Pal, "Genetic algorithm with elitist model and its convergence", *Int. J. Patt. Recog. Art. Intell.*, **10**, 6, pp.731-747, Sept. 1996.

[26] P. Vankeerberghen, "Robust regression and outlier detection for non-linear models using genetic algorithms", *Chemometrics and Intelligent Laboratory Systems*, **28**, 1, pp.73-87, April 1995.

[27] S. Bandyopadhyay, C.A. Murthy, and S.K. Pal, "Genetic classifier using variable string lengths", *Pattern Recognition*, communicated.

[28] P.J. Huber, *Robust Statistics*, 1st Edition. New York: Wiley, 1981.

[29] S. Bandyopadhyay, C.A. Murthy, and S.K. Pal, "Pattern classification with genetic algorithms", *Pattern Recognition Letters*, **16**, pp.801-808, August 1995.

[30] R. Ash, *Real Analysis and Probability*, 1st Edition. New York: Academic Press, 1972.

[31] V. Cherkassky, D. Gehring, and F. Mulier, "Comparison of Adaptive Methods for Function Estimation from Samples", *IEEE Transactions on Neural Networks*, **7**, 4, pp.969-984, July 1996.

[32] B. Mulgrew, "Applying Radial Basis Functions", *IEEE Signal Processing Magazine*, **13**, 2, pp.50-65, Jan. 1996.

[33] R.A. Becker, *The new S language*, 1st Edition. Murray Hill, N.J.: Bell Telephone Laboratories, 1988.

[34] U. Grenander, *Abstract Inference*, 1st Edition. New York: Wiley, 1981.